

The Effect of Organizational Structure on the Adoption of Agile Methodologies: A Case Study

by
Walaa Elshabrawy

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Management Science

Waterloo, Ontario, Canada, 2012

©Walaa Elshabrawy 2012

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This exploratory case study used observations and interviews to investigate how the structure of an organization impacts its ability to adopt agile software development methodologies. It also aimed to identify the agile practices that are perceived as helpful or unhelpful by the individuals practicing them. It examined an organization's attempt to adopt agile methodologies for the first time in a new software product development project. Twelve employees from different teams working on this project participated in the study. . The participants were asked about their perception of the agile process. They were also asked to identify the various teams with which they regularly interact and to provide examples of the helpful and unhelpful patterns of behavior they exhibit.

The findings suggest that the structure of the organization was a major limiting factor that affected its ability to adopt agile methodologies. Agile practices rely on the level of flexibility that an organization can demonstrate. However, the organization attempted to adopt agile practices without redefining the project members' roles, work processes, or departmental affiliations. Participants perceived many aspects of the agile methods negatively, and various symptoms of a misfit between the existing organizational structure and the requirements of agile methods were observed, including poor communication and multiple conflicts between the different project teams, which caused the project to go over time and over budget. Furthermore, it was observed that the teams struggled to follow the agile practices and found various ways to alter and work around them to fit the existing structure, rather than adhering to them and welcoming the new practices.

Several potential areas for future research are identified, including: using longitudinal case studies to examine organizations and the relationships between their members before and after adopting agile methodologies, in order to identify and attribute any observed behavioral patterns to the appropriate cause; examining organizations in which the structure was altered to accommodate agile methodologies; and examining how organizations define the roles of highly specialized employees who possess very specific abilities and must be shared across different development projects.

Acknowledgements

I would like to start by sincerely thanking my supervisor Professor Rob Duimering. Throughout this project, his patience with me, understanding, and encouragement kept me going. I would have been lost indeed without his vast efforts and guidance. I can't find enough words to express my gratitude and appreciation for all of the support and assistance he has provided.

I would like to also thank my readers Professor Frank Safayeni and Professor Kenneth McKay for taking the time to review my work and provide their valuable feedback and comments. I also want to thank my colleagues for their participation in this study and assistance in gathering all the needed information.

Sincere thanks go out to my dear friends Reem and Shady Hassan, Laura and Andrei Radulescu, Ada and Adam Hurst, Christine and Selcuk Onay, Emre Celebi, Bissan Ghaddar, Joe Naoum- Sawaya, and Mehrdad Pirnia for their motivation and support throughout everything.

I will forever be indebted to my parents Ahmed Said Elshabrawy and Karima Abouzied, my brothers Wael and Abdallah Elshabrawy, my sisters Samah Elwakeel and Dina Zayan, and my perfect little nephews Abdelrahman and Ali for their unconditional love and constant encouragement.

Last but not least, I would like to thank my two best friends Khaled Samir and Farah Joukhadar for their encouragement and motivation by asking me every day “aren’t you done yet?”.

Dedication

This thesis is lovingly dedicated to my parents, whose love and support have always, and will continue, to guide me through life.

Table of Contents

AUTHOR'S DECLARATION	ii
Abstract	iii
Acknowledgements	iv
Dedication.....	v
Table of Contents	vi
List of Figures.....	ix
List of Tables	x
1. Introduction.....	1
1.1 Agile Software Development Definition	3
1.2 Agile Software Development History.....	5
1.3 Agile Methods	9
1.3.1 Crystal Methods [4]	9
1.3.2 Dynamic Software Development Method (DSDM) [5]	9
1.3.3 Feature-Driven Development (FDD) [6].....	10
1.3.4 Lean Software Development [7].....	10
1.3.5 Scrum [1].....	10
1.3.6 Extreme Programming (XP, XP2) [8, 9].....	11
1.4 The Distinguishing Factors of Agile Methods	12
1.5 Purpose of the Paper	14
1.6 Outline of the Paper	15
2. Literature Review	16
2.1 Agile Development Literature Review	16
2.1.1 Introduction and Adoption.....	16
2.1.2 Human and Social Factors	20
2.1.3 Perception of Agile Methods	21
2.1.4 Introduction and Adoption.....	23
2.2 Summary	26
3. Method.....	27
3.1 Research Method.....	27
3.2 Interview Format	29
3.3 Participants.....	31

3.4 Analysis and Interpretation of Interview Results	32
4. General Observations	33
4.1 The Organization.....	33
4.2 The Teams.....	34
4.2.1 The Architecture Team	37
4.2.2 The Development Team.....	37
4.2.3 The Functional Testing Team.....	37
4.2.4 The Release testing Team	37
4.2.5 The Operations Team.....	37
4.2.6 The Monitoring Team	38
4.2.7 The Project Management Team:.....	38
4.2.8 The Requirements Management Team:	38
4.2.9 The Documentation Team:	38
4.2.10 The Networking Team:	38
4.3 The Project	39
4.4 The Process	40
4.5 Process Related Observations.....	42
4.5.1 Process Flow.....	42
4.5.2 Tasks Definitions and Durations.....	44
4.5.3 Over-Allocation of Team Members	48
4.5.4 Team Size.....	49
4.5.5 Conflicts	49
4.5.6 Lessons Learned	51
4.6 Summary	53
5. Interview Results.....	54
5.1 Perception of Scrum	55
5.2 Perceptions of the Architecture Team	58
5.3 Perception of the Development Team	60
5.4 Perception of the Functional Testing Team	62
5.5 Perception of the Release Testing Team.....	64
5.6 Perception of the Operations Team	66
5.7 Perception of the Monitoring Team.....	67

5.8 Perception of the Networking Team.....	68
5.9 Perception of the Project Management Team	69
5.10 Perception of the Requirements Team.....	71
5.11 Perception of the Documentation Team.....	73
6. General Discussion and Implications	74
6.1 Limitations	75
6.2 Observations Summary.....	77
6.3 Summary of Interviews.....	79
6.4 Discussion	82
6.4.1 Communication:	82
6.4.2 The Adoption of Agile Methods.....	87
7. Conclusion	91
8. Future Research.....	94
Appendix A - Interview Questions.....	95
Appendix B - Favorable and Unfavorable Aspects of an Organizational Culture	97
References	99
Glossary.....	105

List of Figures

Figure 1 - Teams structure from a process flow perspective 34

Figure 2 - Teams organizational structure..... 35

Figure 3 – Teams structure (abstract)..... 35

List of Tables

Table 1- Agile Manifesto Principals Evaluation (New or not)	8
Table 2 -The Distinguishing factors of Agile Methods.....	13
Table 3 - Allocation example 1.....	42
Table 4 - Allocation example 2.....	43
Table 5 - Over allocation example.....	46
Table 6 - Member #1 allocation.....	46
Table 7 - Member allocation over multiple Sprints	48
Table 8- Team size over multiple Sprints.....	49
Table 9 - Perception of Scrum	55
Table 10 - Perception of the Architecture team	58
Table 11 - Perception of the Development Team	60
Table 12 - Perception of the Functional Testing team	62
Table 13 - Perception of the Release Testing team.....	64
Table 14 - Perception of the Operations team	66
Table 15 - Perception of the Monitoring team.....	67
Table 16 - Perception of the Networking team.....	68
Table 17 - Perception of the Project Management team	69
Table 18 - Perception of the Requirements Management team.....	71
Table 19 - Perception of the Documentation team.....	73
Table 20 - Overall perception of Scrum practices and each team.....	79
Table 21 - Perception of communication	81

1. Introduction

In the software development Industry, it is important to understand the relationship between a company's structure and its ability to adapt to changes in process. Many software companies invest in research in order to identify common failures, so that they can respond to unforeseen scenarios as quickly as possible with minimal impact to their end customers.

Ken Schawber, a leader in the agile software development movement, says, "Methodologies are like cookbooks: follow their recipes and a successful system will result" [1]. This idea implies that agile practices are easy to implement and that they fit any and every company structure. In reality, companies run their businesses in different ways and every environment is not always ideal for the adoption of agile methodologies.

In the 1980s, plan-driven methodologies (like the waterfall model) were commonly used to manage software development projects. By the 1990s, however, one group of developers realized that plan-driven methodologies were dramatically slowing down the software release process. To resolve this, these developers introduced new lightweight software development methodologies [56]. These methodologies utilized only a small number of practices that were easy to follow rather than the existing process-intensive burdensome practices. These new lightweight practices promoted adaptability to design and requirement changes, and they fostered a more collaborative, teamwork-based development process. These sets of practices are known as the "agile methodologies."

Over the next two decades, agile methodologies became very popular. Many software companies quickly adopted them. It was believed that, by following these practices, companies would deliver faster, better and cheaper products. In addition, the shorter release cycle of these methodologies presented great opportunities for companies to adapt to changes in the market. The idea was that they would never lose touch with their end customers and that they could respond as quickly as possible to their customers' changing needs.

Unfortunately, out of a fear of being left behind, a large number of software organizations may have jumped prematurely into the adoption of agile methods. They may have failed to assess the usefulness or fit of these methodologies to their day-to-day processes before implementing them. They may also have failed to identify the impact that adopting new methodologies might have on their organizations' structures and cultures.

This leads us to ask two research questions:

- How does the organizational structure of a company affect its ability to adopt, and how its members perceive agile practices?
- To the individuals within organizations that are adopting agile methodologies, which agile practices are considered helpful and which are unhelpful?

This case study examines the adoption of an agile methodology within a new software development project. The study observes the given project from the moment it was kicked off. Notes and observations are recorded as the different project aspects evolve over the course of its duration. Various teams are interviewed and observed, and conclusions are drawn about how these methodologies were introduced and perceived, as well as how they impacted different teams on the project. A concluding summary describing both the observed benefits and limitations of these methodologies is also provided.

1.1 Agile Software Development Definition

Agile software development is a generic term. It describes a group of software development methodologies that rely on iterative and incremental cycles in order to build and deliver products. A project that uses these methodologies starts with a very basic product idea that needs to be delivered. Then the team builds a product control list. This list contains all of the features that are needed to deliver the end product as well as the tasks that must be performed to achieve this goal. As the project progresses and matures from one iteration to the next, various changes in the features definition and scope take place. These changes result in the constant revision of the control list. As this occurs, the output of the first iteration is fed back into the control list and used to plan the next iteration.

During its lifecycle, the project produces incremental deliverables at the end of each iteration; it delivers a full product by the end of the last iteration. Throughout the development cycle, product features are added, removed, or altered. These feature changes trigger design, requirements and development changes.

Traditional, plan-based methods (e.g., waterfall) follow a sequential design within which a particular phase must be fully completed before the project can move to its next phase. The flow from one phase to another is done in a unidirectional and downwards manner (hence the waterfall analogy). It is not recommended that development teams step back to previous phases after they have been completed.

Plan-based projects typically start with a requirements gathering phase. Changes are not allowed to these requirements after they are marked as completed. At this point, the design phase begins, and this design is based upon the requirements as defined in the previous phase. Given this structure, the existing requirements cannot be modified, and any new requirements cannot be introduced. The project continues to move down through the rest of its phases, from design to implementation, integration, testing, deployment, and finally, maintenance. Any changes that are identified after their corresponding phases are complete are both high-cost and time-consuming.

Agile methods contrast with the traditional plan-based methods in various ways. Instead of following a phased procedure, where one aspect of the product development process is addressed at a time and the output of one phase is delivered right to the next, agile methods allow all of the aspects of development to run at the same time. Product design, requirements, and development phases evolve in

parallel; they interact and feed off of one another. This allows for a faster response time to changes, minimal costs, and low impact to project timelines. It creates enhanced communication between a project's different stakeholders. It also allows stakeholders to rapidly gain better knowledge of the product on which they are working.

1.2 Agile Software Development History

Although agile software development methodologies have gained prominence over the last two decade, it is believed that the roots of agile methodologies go back to the 1970s, if not earlier [40]. Attempts to develop similarly iterative and incremental approaches to development, and to address the perceived problems of more traditional plan-based approaches, have even been made since 1957 [41]. However, these attempts were not taken seriously until a group of developers and consultants decided to put a governing framework around these practices and promote them by means of the “Agile Manifesto” [42].

The “Agile Manifesto” introduced twelve principles. These principles provide high-level recommendations for addressing the problems associated with plan-based software development methodologies. True to the long history of agile methodologies, when the “Agile Manifesto” was introduced, a group of researchers [40] argued that not all of its principles were new. They have provided the following table (Table 1), which examines each of the twelve principles of the “Agile Manifesto” and indicates whether or not they were introduced in previous researches.

Principle	New or Not, with Evidence
Customer satisfaction through early and continuous delivery of valuable software.	Not new: The first principle of Evolutionary Development (EVO) states “deliver something to the real end-user” [46].
Welcome changing requirements, even late in development.	Relatively new: This problem always existed but did not have a real solution.
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter timescale.	Not new: In EVO, frequent and early delivery is essential. Rapid Application Development (RAD) also recommended quick delivery [46, 47].

Business people and developers must work together daily throughout the project.	Relatively new. Some previous approaches recommended good relations with customers, but the ideas of daily communication and on-site customers are new.
Build projects around motivated individuals.	Not new: These ideas were raised in a book on the psychology of computer programming that was published in 1985. The author emphasized the importance of motivation, which is the inner, directing force. In addition, he mentioned that the richness of an environment gives it a self-maintaining quality that resists imposed changes [51].
Convey information to and within a development team through face-to-face conversation.	Not new: The aforementioned book focused on the importance of how the working space can affect social interactions, which in turn affect the work. The author emphasized how face-to-face communication helps in transmitting useful information [51].
Working software is the primary measure of progress.	Not new: The second principle of EVO states “measure the added value to the user in all criteria dimensions” [46].

The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	Not new: In <i>Death March</i> , Edward Yourdon pointed out the importance of managing and controlling progress. he put forth the “daily build” concept to succeed in that mission [52].
Continuous attention to technical excellence and good design enhances agility.	Not new: A sense of the importance of doing a much better programming job (i.e., technical excellence) can be found in Dijkstra’s famous article “Humble Programmer” [53].
Simplicity—the art of maximizing the amount of work not done—is essential.	Not new: The famous saying about the simplicity of design comes from Antoine de Saint-Exupery: "Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away"[49].
The best architectures, requirements, and designs emerge from self-organizing teams.	Not new: The idea of the self-organizing team appears in open source projects that came out at roughly the same time as the “Agile Manifesto.” In “The Cathedral and the Bazaar” paper, Raymond referred to developers as people who bring their own resources to the table [48].
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	Relatively new: The idea of process improvements was presented in level 5 of the Capability Maturity Model Integration (CMMI), but with different emphasis than it has in agile

	work. The idea was that all of the team, and not only the management, should reflect on improving the process [50].
--	---

Table 1- Agile Manifesto Principals Evaluation (New or not)

1.3 Agile Methods

Various agile methods are currently used in the software development industry. These methods share the basic idea of following an iterative and incremental process during which all of the aspects of product development run in parallel. However, these methods vary greatly from the point of view of their implementation. They follow different sets of practices and have their own terminologies and tactics. The most commonly used agile methods [3] are listed and defined below.

1.3.1 Crystal Methods [4]

The Crystal methodology is one of the most lightweight agile methods. It comprises a family of methodologies (Crystal Clear, Crystal Yellow, Crystal Orange, Crystal Orange Web, Crystal Red, Crystal Maroon, Crystal Diamond, and Crystal Sapphire). The Crystal family focuses on communication in small teams (of ten or fewer members). It introduces a set of practices and processes that can be tailored to address different levels of complexity and criticality. Like other agile methods, Crystal methodologies promote the early and frequent delivery of working software as well as customer involvement and adaptability.

1.3.2 Dynamic Software Development Method (DSDM) [5]

The Dynamic software development method provides an independent framework that can be used alongside other agile methods. This framework is used as a foundation for planning, managing and executing agile development projects. It divides projects into three phases: pre-project, project lifecycle and post-project. It relies on nine key principles that revolve around user involvement, empowering the project team, frequent delivery, addressing current business needs, incremental development, reversible changes, integrated testing, and effective communication.

1.3.3 Feature-Driven Development (FDD) [6]

The feature-driven development methodology combines model-driven and agile development with an emphasis on the initial system model. Once a model is in place, the list of features to deliver is defined; then each feature is addressed in an iterative fashion. An iteration of a given feature consists of two phases: design and development. Iterations are short (i.e., two weeks, and if a feature is too large to be delivered in two weeks, it is split up into smaller, more manageable features). Development teams deliver project features by using the following eight practices: domain object modeling, developing by feature, component/class ownership, feature teams, inspections, configuration management, regular builds, and visibility of progress and results.

1.3.4 Lean Software Development [7]

This method is an adaptation of principles from lean production and in particular from the Toyota production system. This method focuses on selecting only the features that are valuable for a system. These features are later prioritized and delivered in small iterations. The speed and efficiency of the development process is also a main focus of this method as it relies on continuous and reliable feedback between developers and end customers. Lean software development relies on the following seven key principles: eliminating waste, amplifying learning, deciding as late as possible, delivering as fast as possible, empowering the team, building integrity, and seeing the whole.

1.3.5 Scrum [1]

Scrum is a lightweight management framework that can be applied to various types of projects. It is a self-managed, list-driven, iterative process that focuses on project management in situations where it is difficult to plan ahead. The lifecycle operates as a sequence of fixed-length iterations, called “Sprints,” during which a small subset of features is selected from the main list, called the “Product Backlog,” and fully developed in that timeframe. Each Sprint is planned when the iteration starts and reviewed when it ends. One key deliverable from each Sprint is a demonstration of the completed features. These demonstrations are held primarily for the stakeholders, but for any other interested parties as well. A

Sprint typically lasts two to six weeks and is tracked via very brief daily meetings – called the Stand-up meetings - among the project team.

1.3.6 Extreme Programming (XP, XP2) [8, 9]

XP is a disciplined approach to delivering high-quality software both quickly and continuously. In XP, the end customer works closely with the project team to define the project's main requirements in the form of "User Stories." The project team estimates the effort that will be needed to deliver each user story and prioritizes them. The User Stories ensure that the defined features are added incrementally to the working product, which is delivered at highly frequent intervals. XP relies on the following twelve key principles: planning game, small releases, customer acceptance tests, simple design, pair programming, test-driven development, refactoring, continuous integration, collective code ownership, coding standards, metaphor, and sustainable pace.

1.4 The Distinguishing Factors of Agile Methods

The table below (Table 2) highlights the different unique factors that distinguish each agile method.

Agile Method	Distinguishing Factor
Crystal	<ul style="list-style-type: none">• Dependent on the size of the project team• Dependent on the criticality of the project• Employs minimal process and extends this process only when necessary
Dynamic Software Development (DSDM)	<ul style="list-style-type: none">• Provides the possibility of reverting to an older version of the product• Provides early delivery of the business benefit and feasibility of the product
Feature-Driven Development (FDD)	<ul style="list-style-type: none">• Can be scaled to accommodate larger teams• Features are delivered in two-week iterations
Lean Software Development	<ul style="list-style-type: none">• Delays making decisions as much as possible until enough facts are available• Identifies product interaction with other systems and not just the functionalities that the product needs to deliver
Scrum	<ul style="list-style-type: none">• Intended for teams of seven members (plus or minus two)• Daily Stand-up meetings

	<ul style="list-style-type: none"> • Burn down chart to display progress • Features are typically delivered in 30-day iterations
Extreme Programming (XP, XP2)	<ul style="list-style-type: none"> • Intended for teams of 10–12 members • Code refactoring through frequent code testing to ensure that no deeper code problems exist • Pair programming • Enforces coding standards

Table 2 -The Distinguishing factors of Agile Methods

1.5 Purpose of the Paper

With the term “agile” being a buzzword for a number of years in the software development industry, many researchers were encouraged to take a closer look at the practical application of agile methodologies in real life situations. They have examined their benefits and limitations from various aspects in attempts to provide the software development community with guidance on how to maximize their benefits by correctly applying them. Such researches can help organizations in choosing the agile methods that fit them the most. They will help them as well in knowing when, how, and where in the organization to apply them.

The purpose of the present study is to extend this research on agile methodology adoption and application within the software development industry. This study intends to examine the challenges and limitations that surface when organizations implement these methodologies for the first time. It will focus on assessing the impact of an organization’s structure on the adoption process. The specific benefits and challenges observed in this case study will be discussed, and conclusions will be drawn and recommendations will be made where appropriate.

1.6 Outline of the Paper

This study is divided into the following sections:

- a literature review of the various studies done on agile methodologies with a specific focus on research done on their adoption and introduction;
- a description of the study design. This will include the method used to gather the data, the participants, and the structure of the questionnaire that was used in this study;
- the different trends and patterns observed during the study;
- the interview results and analysis;
- a conclusion about the study along with a discussion of the study's implications;
- limitations of the study and suggestions for future research; and
- a general summary.

2. Literature Review

2.1 Agile Development Literature Review

In a recent literature review on agile software development, Dyba and Dingsoyr [3] identified 1996 studies that examined the impact of the adoption of agile methods within various sectors. This review focused eventually on 30 empirical studies that were categorized into the following four groups: introduction and adoption; human and social factors; perceptions of agile methods; and comparative studies. The section below summarizes the review findings in each of these categories

2.1.1 Introduction and Adoption

Several studies have examined the first-time introduction and adoption of agile methods in various organizations. These studies covered three main topics: introduction and adoption; the development process; and knowledge and project management.

2.1.1.1 Introduction and Adoption

Bahli and Abou Zeid [10] examined two projects, TrackMed and InsightMed that were developed within the same organization. TrackMed was developed using a waterfall process, while InsightMed was developed using an XP model. Both systems provided similar functionalities. The company concluded that the adoption of XP practices enhanced knowledge creation among the different teams that were involved in the development process. It also concluded that XP was well received by the development team, and it intends to use XP for all of its future projects.

The authors highlight a few drawbacks to the use of XP. The first has to do with the management of project iterations. It was observed that each iteration took considerable amounts of time to plan, to assess what had been achieved once it was completed, and then to use these findings in order to plan for the next iteration. This more administrative part of the process consumed a big chunk of the project manager's

time. The second drawback was related to reporting the progress of the project to the management team. Because the iterations were repetitive, it was difficult to convey to the management team whether or not the project timeline was on- or off-track.

Additional factors are worth noting. The project that was developed using waterfall (TrackMed) was begun much earlier than the project that was developed using XP (InsightMed). By the time that TrackMed was delivered, it was almost 50 percent over budget and 60 percent over schedule. On the other hand, when the XP-led project was begun, the teams already had a good understanding of the project's requirements as a result of their prior experience with TrackMed. They were also able to rely on one of TrackMed's existing components in building the second project. In short, the development of InsightMed (by means of XP practices) was not exposed to the same conditions as the development of TrackMed (by means of waterfall). This may be one factor in why the XP methodology was well received by the development team, given that a great part of the ambiguity and uncertainty related to the project's details had been removed prior to its adoption.

Another factor in assessing the reliability of these findings is that the XP project had a relatively higher code quality as delivered but was still 25 percent over budget and 50 percent over schedule. Also, not all of the features that were initially requested were delivered as part of the final product.

In summary, the adoption of XP seems to have translated into some enhancements to the development process, but it did not fix some of this organization's existing problems. The project was still over budget and past its deadline when the team delivered the final product. Furthermore, given its missing features, the delivered product was not the complete product for which the organization initially aimed. It can be argued that the enhancements that the development team perceived may be attributed either to the use of XP or to the team's familiarity with the problem domain. This leads us to ask whether using XP practices is more convenient when extending an existing product versus the development of a product from scratch.

In another study, Svensson and Host [36] examined the introduction of XP practices for a project that focused on the maintenance of existing products. Some of the introduced practices were adapted to fit

the existing development environment. Additionally, certain XP practices were not introduced at all because they did not affect the team's overall development process.

In this study, it was concluded that the introduction of XP practices was difficult because of the complexity of the organization studied. It was recommended that an assessment of the organization's software development processes should be done prior to the introduction of any agile process. This assessment allows organizations to understand what, where and why any new process should be introduced,

Like the previous study [36], J. and N. Livari [55] indicate that an assessment of organizational culture is needed prior to the introduction of XP. Their study focused mainly on identifying various aspects of organizational culture that favorably or unfavorably influences the adoption process. The authors studied seven cultural dimensions of organizations in six different companies. These dimensions were: innovation and risks, detail orientation, outcome orientation, people orientation, team orientation, aggressiveness, and stability. They compiled a list of favorable and unfavorable aspects [Appendix B], which indicate whether the culture of a given organization is suitable to the adoption of agile methodologies.

A final study examined a set of hypotheses about assessing the relationship between four organizational culture dimensions and the adoption of agile methods. In this study, Tolfo and Wazlawick [54] used the Competing Values Model (CVM), which distinguishes between stability and change as well as between internal and external focus in order to define the dimensions of an organization's culture. They conclude with a number of interesting points. Notably, the authors found that agile methods are highly incompatible with an orientation toward a hierarchical culture. If agile methods are adopted within an organization with a relatively strong hierarchical culture, the organization should combine complementary features of different agile methods, such as XP and Scrum. This makes these combined models heavier. As a consequence, the methods may lose some of their emergent agility.

2.1.1.2 Development Process

Several studies have examined the impact of agile methods on the development process. These studies mainly focused on XP methods. In two studies [35, 37], it was concluded that having the customer on-site provided better results. Tessem [37] examined pair programming versus single programming. In this study, pair programming was well received by a group of developers, while another group found it to be “very exhausting” and “a waste of time.” Towards the end of the project examined in this study, the development team shifted towards single programming.

In another study, Hilkka, Tuure and Matti [16] concluded that XP methods worked best among experienced developers with domain and tool knowledge. In a third study, Middleton [29] examined XP adoption by two two-person teams. In Middleton’s study, one team was able to make fewer errors with every new iteration, while the other team continued to make a high number of errors. He concluded that, “by moving the responsibility for measuring quality from the manager to the workers, a much quicker and more thorough response to defects was obtained.”

2.1.1.3 Knowledge and Project Management

Among the introduction and adoption literature, several studies that focused on knowledge and project management were examined. Bahli and Abou Zeid [10] concluded that the adoption of XP practices improved knowledge sharing. This was due to the shorter cycles, constant communication, and closer interaction between team members that characterize XP practices. Another study focused on the feasibility of applying agile methods in large projects. Here, Karlstrom and Runeson [18] observed that when engineers raised technical problems early on to their managers, problems occurred due to a conflict of perceptions between the engineers and their managers.

In a third study, Tessem [37] examined the work-size estimation capabilities of a series of XP teams and concluded that these estimates improved as the projects approached completion. The participants noted that not enough design and architecture discussions occurred throughout the project. In a final study, Svensson and Host [35] concluded that XP practices had a positive effect on collaboration within the company that was studied.

2.1.2 Human and Social Factors

Several studies that focused on the human and social factors that are associated with the adoption of agile methods were examined. These studies covered the three main topics of organizational culture, collaborative work, and team characteristics.

2.1.2.1 Organizational Culture

Robinson [32] observed three different companies in the UK. He concluded that although the three companies varied in organizational type, structure and physical setting, XP methods were well perceived and worked well for all three.

2.1.2.2 Collaborative Work

Several studies were examined that focused on one or more of the following aspects: the role of conversation in collaborative work; how progress is tracked; and how work is standardized. In one of the studies, Robinson and Sharp [31] focused on conversation and described pairing as intense and stressful. In a second, Mackenzie and Monk [22] emphasized the importance of conversation. They also concluded that the planning process “spans the usual boundaries between project managers and software developers”. J. Chong [13] described progress tracking as happening on the two levels of “daily rhythm and the rhythm oriented around the iteration.”

2.1.2.3 Team Characteristics

Several studies that focused on the characteristics of the teams who adopted XP methods were also examined. In one study, Robinson and Sharp [30] concluded that “these teams have faith in their own abilities, show respect and responsibility, establish trust, and preserve the quality of working life.”

In a second study that examined those personality traits that, when exhibited by team members are perceived by them as beneficial. Young, Edwards, McDonald, and Thompson [39] concluded that good XP team members are “analytical, with good interpersonal skills and a passion for extending [their] knowledge base and passing this on to others.”

2.1.3 Perception of Agile Methods

Several studies that focused on the perception of agile methods by different groups were examined. These studies covered the three topics of customer, developer, and student perceptions,

2.1.3.1 Customer Perceptions

Several studies that focused on customer perceptions of agile methods were examined. Each of these studies addressed customer satisfaction and/or customer role and collaboration with development teams.

Two studies concluded that the use of agile methods led to high customer satisfaction. Ilieva, Ivanov and Stefanova [17] discovered that this occurred because the customers felt that they had good control over the development process. Mann and Maurer [23] concluded that customers felt that they were kept informed through daily meetings, which led to less confusion about what was being worked on and delivered.

As for the issue of customer role and collaboration, Martin, Biddle and Noble [25] concluded in a single study that the customer worked long hours and was under continuous stress. Koskela and Abrahamsson [19] calculated that the customer spent most time on planning sessions and acceptance testing followed by retrospective sessions. Another study focused on the customer role in outsourced

projects. In this study, Martin, Biddle and Noble [26] concluded that these projects were challenging because the customer had to familiarize itself with the culture of the developers and their organization.

2.1.3.2 Developer Perceptions

A number of studies that focused on developer perceptions of agile methods were also examined. One study used a web-based questionnaire in order to survey job satisfaction level among employees who work in companies that have adopted XP methods and those who work in companies that have not. In this study, Mannaro, Melis and Marchesi [24] concluded that employees who work for companies that have adopted XP methods have higher levels of job satisfaction and feel more comfortable and more productive in their jobs. In another study, Ilieva, Ivanov, and Stefanova [17] concluded that pair programming is “a very useful style of working as everyone was strictly conforming to the coding standards.” In a third study, Mann and Maurer [23] examined Scrum adoption and concluded that the adoption of Scrum led to overtime reduction, as well as that developers were satisfied with both product quality and customer involvement. In a final study, Bahli and Abou Zeid [10] focused on XP adoption in a company that develops medical information systems and concluded that the developers at that company perceived XP methods as being both useful and easy to use.

2.1.3.3 Student Perceptions

Two studies that focused on student perceptions of agile methods were examined. In these studies, Melnik and Maurer [27, 28] found that the 240 students that were surveyed in both studies at the University of Calgary and the Southern Alberta Institute of technology were “very enthusiastic about core agile practices.”

2.1.4 Introduction and Adoption

Several studies that compared agile methods to alternative methodologies were examined. These studies covered the four topics of project management, productivity, product quality, and work practices and job satisfaction.

2.1.4.1 Project Management

Several studies that compared the project management dimension of agile methods to various alternative methodologies were examined. Ceschi, Sillitti, Succi and De Panfilis [12] surveyed both plan-based and agile companies and concluded that the use of agile methods improved the software development process as well as the customer relationship.

Chong [11] stated that for companies that adopt agile methods “projects do not begin or end, but are ongoing operations more akin to operations management.” In a third study, Karlstrom and Runeson [18] observed that the incorporation of agile methods for day-to-day work with the simultaneous use of a stage-gate project management technique led to better results. These results included better communication, better tools to manage day-to-day work, as well as better reporting.

Dagnino, Smiley, Srikanth, Anton and Williams [14] compared a project that used an evolutionary agile approach and another that used a more traditional incremental approach. They concluded that the agile method assisted in the easy adaptation to requirement changes in the later stages of the project and with less overall impact than the traditional approach. They also observed that the agile approach demonstrated business values more quickly and with more frequent releases.

In a final study, Sillitti, Ceschi, Russo and Succi [34] compared agile methods to traditional plan-based methods and concluded that companies that use the former are more flexible and customer-centric, which in turn leads to higher customer satisfaction. With respect to human resource management, however, Baskerville, Ramesh, Levine, Pries-Heje and Slaughter [11] concluded that, “compared to traditional development, team members of agile teams are less interchangeable and more difficult to describe and identify.

2.1.4.2 Productivity

Several studies that compared the productivity of development teams using agile methods with the productivity of teams using plan-based methods were also examined. These studies documented the number of Lines of Code (LOC) that were produced by developers using each method and then calculated gains in percentage of productivity.

Dalcher, Benediktsson and Thorbergsson [15] concluded that the observed developers who used agile methods displayed a productivity gain of 337 percent. They have also showed that XP teams delivered 3.5 times more lines of code than V-model teams. Macias, Holcombe and Gheorghe [21], on the other hand, reported that there was no difference in product size between XP teams and traditional ones.

While in a second study, Ilieva, Ivanov and Stefanova [17] showed a smaller gain of 42 percent. In a third study, Layman, Williams and Cunningham [20] showed a productivity gain of 46 percent, and a final study, conducted by Wellington, Briggs, and Girard [38], showed a productivity *loss* of 44 percent.

Other studies focused on the productivity related to agile methods. Mannaro, Melis and Marchesi [24] tried to assess whether or not a productivity increase was perceived by developers as the result of the development process choice. They showed that “on a scale from 1 (Strongly Disagree) to 6 (Strongly Agree), the mean for the non-XP developers was 3.78, while the mean for the XP developers was 4.75.” In a second study, Melnik and Maurer [28] came to similar results by showing that 78 percent of study participants believed that the use of XP methods increased the productivity of their small teams.

2.1.4.3 Product Quality

Several studies have examined the quality of products that were released as the result of agile methods versus products that were released as the result of more traditional methods. Layman, Williams and Cunningham [20] showed that, among releases involving agile methods, there was a 65 percent

improvement in pre-release quality and a 35 percent improvement in post-release quality for newer releases of a particular product.

In a second study, Ilieva, Ivanov and Stefanova [17] showed that 13 percent fewer defects were reported in XP-developed projects versus non-XP-developed ones. In a third study, Wellington, Briggs, and Girard [38] concluded that the actual code of projects involving XP scored better than that of non-XP projects on the quality metric used. A final study examined ten XP teams and ten traditional teams in order to measure the internal and external quality of the code developed by each group. Interestingly, in this study, Macias, Holcombe and Gheorghe [21] reported that there was no difference in quality between the groups.

2.1.4.4 Work Practices and Job Satisfaction:

Several studies that made qualitative comparisons of social behaviors were examined. Chong [13] compared the work routines and practices of XP and non-XP teams. He concluded that because XP makes these practices more visible by the team members, it provides a more standardized way of performing development tasks.

In another study, Wellington, Briggs and Girard [38] examined team cohesion and individual attachment to projects in both XP and non-XP teams. Their study showed equal or more cohesion between members of non-XP teams than that displayed by XP teams. However, the study also revealed a general lack of cohesion in the sub-teams within the non-XP teams.

The last study examined job satisfaction among members of XP and non-XP teams. In this study, Mannaro, Melis and Marchesi [24] showed that XP teams are generally more satisfied with their jobs and more comfortable in their work environments.

2.2 Summary

The literature review [3] summarized in this section addressed a number of different aspects of the introduction and use of agile methods within the software development industry. Scholars who research the field of software development have attempted to understand how the structure of a certain company affects its ability to adapt to process changes. Many studies have been conducted to assess the impact that the culture and characteristics of organizations have on the adoption of new processes. However, the majority of these studies focused on examining projects teams and their interactions in isolation from the organizational structure and its impact on the adoption process. This represents a gap in the field.

This study aims to address this gap by means of a case study that examines the first-time introduction of agile practices to an organization. It will observe how these newly introduced agile practices altered existing company structure, or had to be tailored to fit this structure. Through interview results and observations, interpretations and conclusions about this topic will be presented. Finally, the study will present a summary that discusses the limitations that company structure may be imposing on the adoption process.

3. Method

3.1 Research Method

From the beginning, it was decided that this study was exploratory. No specific hypotheses were developed and no prior expectations were identified. Instead, the study's purpose was to closely and continuously observe different development teams and to determine how these teams adopted and perceived agile methods when the methods were first introduced. These teams had to deal with a shift from traditional to agile practices. This shift was easy for some members and difficult for others. Individual members also had to deal with conflicts that arose as a result of this shift, and they had to face various changes in overall development processes, which had an impact on the nature of their day-to-day tasks.

It became apparent that this research needed to be approached using a qualitative exploratory method rather than a quantitative one. Quantitative methods are convenient when researchers have a good understanding of the area that they wish to research before they do this research, while qualitative research is more appropriate when researchers seek to analyze and understand a new situation.

Furthermore, when observing the project this study is examining, the areas where this shift in software development practices would impact were not obvious at all at the beginning. With no expectations and no hypothesis, it was hard to know which data to record, which interactions to focus on, and what type of observations would eventually be relevant since it was not clear yet how it will all come together. A decision was made to record as many details as possible as the case evolved. An assumption was made that changes will occur in the operational realm (end product quality, the length of the release cycles, etc...), as well as in the personal realm (relationships between different teams, job satisfaction, etc...)

A few factors helped greatly in narrowing down the scope of items on which to focus and in determining which activities to record. These factors were:

1- Being part of the project team

Being an employee at the company where this research was conducted facilitated the gathering and observing of various details. Additionally, being a requirements analyst for the project that was examined in this research gave me firsthand experience of the impact of this shift on the requirements team as well as its impact on other teams. Being part of the environment before and after agile adoption allowed me to gain a better and deeper understanding of the specific changes that took place within the organization. I was also able to understand the importance of these changes, which made it easier later to determine on which changes to focus. Finally, the understanding of the project's scope and structure that I gained as a technical project member assisted me greatly in knowing which questions to ask, to whom to direct these questions, and what the various project members meant by their use of certain terms.

2- Availability of direct and indirect resources

While the majority of the data gathered in this study was provided by questionnaires that were completed by project team members, off-the-record conversations and friendly discussions provided further details, especially when conflicts arose. This allowed me to understand the different perspectives that each of the project members adopted. Also, having access to official records like project management documentation, different teams' deliverables, Sprints, and retroactive files allowed me to determine when the comments made by a project member were supported by official records and when there was a mismatch of information.

3- Personal experience

Working in the software development industry for the last eight years has exposed me to both plan-based and agile methods. This experience has made it easier for me to understand the different terms used about these methods. It has also allowed me to see instances when "official" practices were applied and followed as well as situations within which altered versions of these practices were used.

3.2 Interview Format

Twelve participants were interviewed individually and in person for a total of twelve interview sessions. These interviews were conducted over the span of several weeks near the end of the code development phase. The sessions lasted between approximately 20 and 50 minutes.

All of the interview sessions followed the same format. I started by explaining the objective of this research to the participants and obtaining their consent to record the interview session with a digital recorder. At the beginning, a few examples of questions that were to be asked were provided and the participants were informed that they had the right not to answer any question if they so choose, as well as to terminate the interview session at any time. Once the interview began, the participants were asked a specific and predetermined set of questions, one at a time. Participants were allowed to provide their answers uninterrupted. Sometimes a question had to be rephrased if it was sensed that a participant did not fully understand it. Additionally, guidance was sometimes given to participants to ensure that they did not digress from the topic under discussion.

The interview questions (Appendix A) were divided into four sections. The first section included questions about participants' current positions, how long they had been in those positions, and how long they had been with the company overall. Participants were also asked to briefly describe their daily activities and to define their day-to-day tasks. This set of questions aimed to gather information about the participant's current status and experience with following the previous waterfall and current agile practices in their daily activities.

In the second section, the participants were asked about their involvement with the project under observation in this study. They were asked about their workload on that project, time spent on meetings, time spent on daily tasks, their perceptions of the usefulness of meetings, and other related activities. These questions provided a deeper understanding of the practices followed by the participants and how their daily activities were impacted by them.

The third section explored the participants' relationships with other teams. These questions asked them to identify the various teams with which they regularly interact and to provide examples of the helpful and unhelpful activities performed by each of these other teams. The participants were encouraged

to provide as many examples as possible and to focus on the professional aspects of their interaction with other teams rather than on more personal interactions. These questions stepped further in exploring the impact of the newly introduced practices on the relationships between different teams.

Finally, the fourth section focused on gathering information related to how participants viewed the overall experience of working on their first agile project. They were asked whether they had observed any conflicts during their involvement with the project, how these conflicts could have been avoided, what they saw as the lessons learned from this experience, and what changes they hoped to see made for future projects. This final set of questions provided more insight into the areas that were negatively perceived by each of the project members.

3.3 Participants

The twelve participants constituted a subset of the larger group of stakeholders who are assigned to develop new products from scratch. The participants included ten males and two females with varying educational levels and with occupations that represented the project's different teams. The numbers of years each participant had been with the company ranged from eight months to ten years. Participants' experience with agile methods ranged from none to very experienced.

These twelve participants were recruited because they represented the different teams that were involved in the project under consideration. This ensured that different points of view were considered in the study and that the impact of the application of agile methods across various teams with different dynamics was also assessed.

3.4 Analysis and Interpretation of Interview Results

The recorded interviews were transcribed verbatim into text documents that were refined through different iterations. The researcher began with listening to the recorded interviews and transcribing each word by word into a text document. The files were then analyzed by going through each interview and reading the interview questions and answers, thus ensuring that the answers that were provided did in fact belong to the given question. If information relating to a different question was provided, this information was then moved to the appropriate question.

The files were examined for a second time in order to summarize each participant's answers to the questions. Then, the researcher reviewed all of the summaries in order to group together the participants' answers to a given question and to assign these answers to distinct categories. Finally, these categories were themselves analyzed in order to identify common remarks and patterns of responses. This process allowed similar ideas and information to explain what was happening to be grouped together. Also the number of participants in each category was counted as an indication of participants' agreement.

4. General Observations

In order to understand the results of this study, it is important to understand the circumstances surrounding the study as well as the nature and internal culture of the company in which it took place. The following notes were drawn from my personal observations of the different events that took place during the development of the observed project. These observations were also based on my experience as a full-time employee in the functional area where this project was executed.

4.1 The Organization

This research was conducted in a large telecommunications company with about 16,000 employees worldwide. This company was chosen because of convenience and opportunities for access that were the result of my being a full-time employee. Additionally, the fact that the teams working on the project under observation were new to agile methodologies made this choice more appealing. A radical change occurred when these teams decided to change their project management system from a traditional waterfall to an agile system, and this was worth observing and investigating. It is worth noting that the agile methodology was implemented in other areas within this organization. However, given the large size of the organization as well as the segregation between different departments from an operational perspective, the department where this research was conducted had no prior exposure to implementing their projects in an agile fashion.

4.2 The Teams

The teams involved in executing this project were responsible for a particular area within the company that deals only with internal software products. The internal products that they developed were part of the software infrastructure that supported the company's external products as well as services that were used by end customers. The sizes of the teams ranged from 5 to 18 members. However, for this project only a few representatives from each team were directly involved

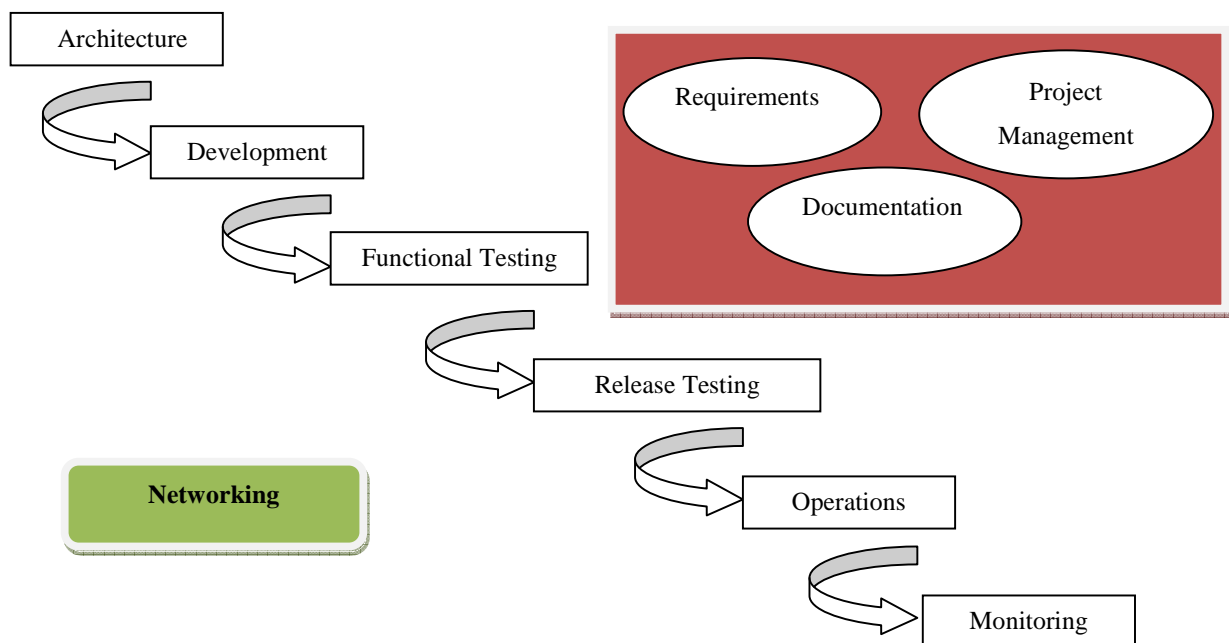


Figure 1 - Teams structure from a process flow perspective

Figure 1 shows the structure of the project teams from a process flow perspective, while Figure 3 shows the organizational structure of each team. Figure 2 is an organizational chart of the company and highlights the reporting hierarchy for each department

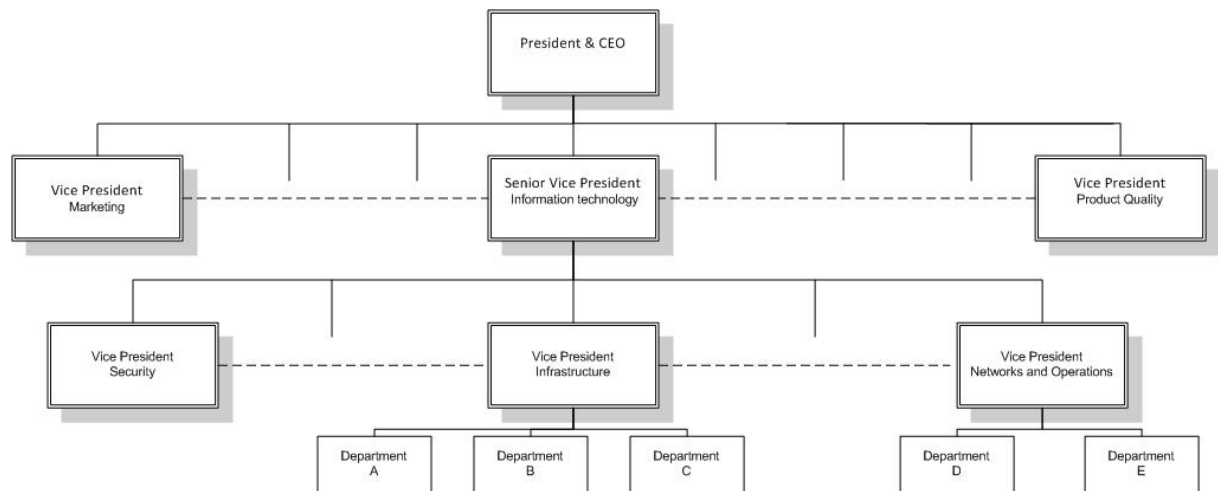


Figure 2 - Teams organizational structure

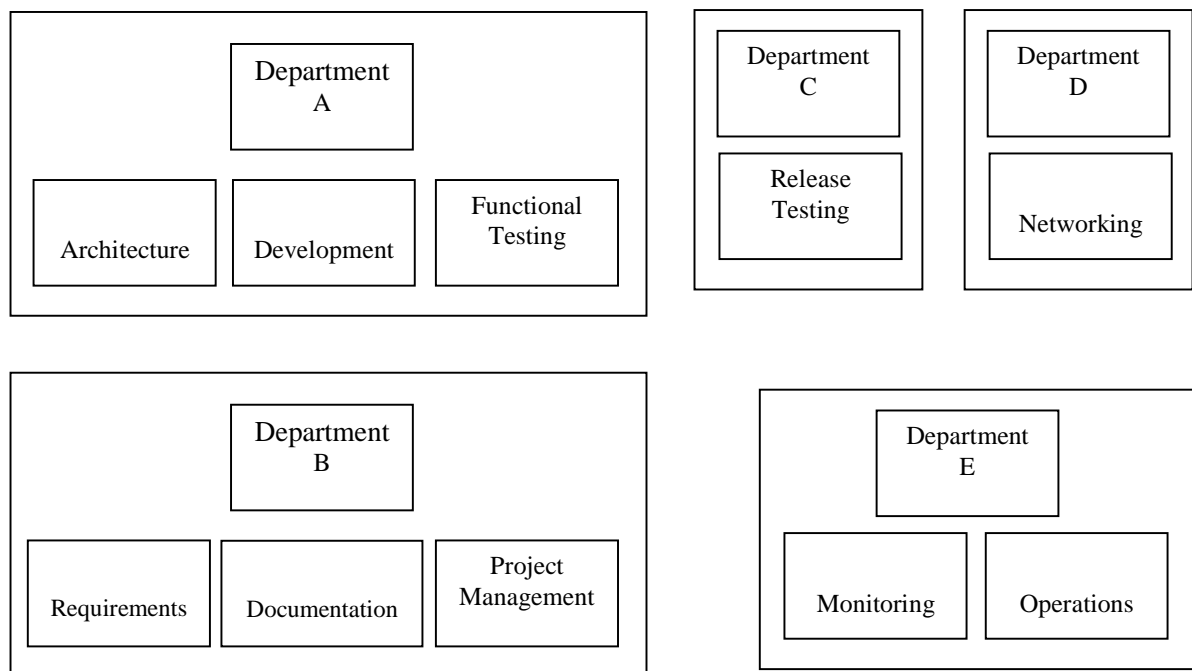


Figure 3 – Teams structure (abstract)

From a process flow perspective, any new internal product as well as any change to an existing internal product has to flow through the different teams in the order shown in Figure 1. Originally, the teams were

arranged in a sequential, waterfall-like structure within which each team worked on the product at a given time and then handed the product off to the next team downstream. There were occasions where two consecutive teams worked in parallel; however, these were uncommon.

From an organizational perspective, these teams belonged to different areas or departments, as shown in Figures 2 and 3. Furthermore, as can be seen in the organizational structure, each team (or group of teams) reported to a different management team. Different teams also followed different internal processes. For example, status reports were done via weekly emails in some teams. While in others, they had a weekly or bi weekly in-person meeting with their superior to discuss their work load and progress. The same also can be seen when it came to each team's deliverables. Some teams had a single deliverable that is published for other teams to use. Others had multiple deliverables, with each provided to one or more team.

These teams were not new to each other when they started to work on this project. They had worked together previously on multiple projects and had developed patterns of interaction along with perceptions of each other's competency and professionalism. Within the teams, members appeared to get along very well with each other and to have reached general agreement about most of the decisions made by the teams.

Within these projects, decisions and changes related to the project tended to follow the same virtual waterfall-like functional structure where the teams upstream possessed more control over the formulation and development of the product than those downstream. It can be argued that this is what should have happened, given that upstream teams are the ones who build the actual product, but the downstream teams felt that they needed to have input into the initial phases of the product development cycle. One of the participants from a downstream team said, for instance:

“When you are five levels above production, you will miss something because you are not exposed to the day-to-day issues like the downstream folks.”

Whether intentionally or not, the virtual organizational structure shown also triggered certain perceptions of the participants on different teams and assisted in shaping the dynamics of the interactions

and relationships among them. There was the general perception that the level of control one had on a particular product or project was defined by the team to which a person belongs.

Below is a description of the teams involved in the project under observation.

4.2.1 The Architecture Team

This team was responsible for designing internal products and deciding which features were needed for their development. The architecture team was considered the product's owner and had the authority to prioritize work for other teams.

4.2.2 The Development Team

This team was responsible for translating the product's design aspects into code and ensuring that new features were properly integrated into the existing system without impacting other functionalities.

4.2.3 The Functional Testing Team

This team was responsible for testing the developed code and ensuring that the delivered functionalities matched those that had been specified in design documents.

4.2.4 The Release testing Team

This team was responsible for testing how the overall product was to be released and integrated with the existing infrastructure as well as how the product settings would be configured before it is available to the end customers.

4.2.5 The Operations Team

This team was responsible for where, how, and when to release the product.

4.2.6 The Monitoring Team

This team was responsible for monitoring the released product and ensuring that it worked properly. This team was also responsible for resolving or escalating any emergencies that arose.

4.2.7 The Project Management Team:

This team was responsible for managing the overall project in order to ensure that every team provided its required deliverables within specified times and budgets.

4.2.8 The Requirements Management Team:

This team was responsible for gathering the product requirements from different stakeholders. These requirements then acted as a contract that specified which features would be delivered as well as which changes would be needed to accommodate these new features.

4.2.9 The Documentation Team:

This team was responsible for documenting the product code, its functionalities and its configuration settings.

4.2.10 The Networking Team:

This team was responsible for making any necessary changes to the company's network to support the release of new products.

4.3 The Project

The intent of the project under observation was the development of a new internal product. From an architectural perspective, this product introduced major changes to the company's existing infrastructure system. The size of this project was considerably large in comparison to existing and previous ones. The infrastructure system was responsible for providing the base on which the end customer facing products run. These architectural changes meant that all of the teams in this functional area had to change their processes and procedures in order to adapt to the new system.

When this project was initiated, it was introduced to the project teams as a research initiative. This research initiative was intended to only assess the feasibility of the new design and to provide recommendations as to whether or not the product was worth implementing. During the first few months of this initiative, it was decided that the project would not be a research initiative anymore, but would instead be executed as a regular and full-blown project with deliverables that would be implemented and integrated within the company's existing system. The initial project plan, which was created in May 2010, indicated that the project would be completed within a twelve-month timeframe. It is worth noting that, at the time of writing (July 2012), the project had not yet been completed.

4.4 The Process

At the beginning, the upper management team made the decision to use the project as a pilot project for testing the introduction of the Scrum agile methodology to this particular department. Prior to this, all previous projects were executed using a traditional waterfall method. The shift from using waterfall to a Scrum agile methodology introduced many changes and forced different teams to adapt.

Prior to adopting agile practices, the different teams involved in this project followed traditional waterfall methodologies in their software development process. The process was run in a fashion similar to a factory production line where one team works on the project in a particular phase then hand it off to the next downstream team. With this type of process, a project's members were only involved in meetings during the time in which they were actually working on the project. Short hand off and knowledge transfer sessions were held as needed as the project moved from one team to the next. Once a team was done with their work in a particular project, their involvement with this project was kept to the minimum.

With the adoption of agile methodologies, the project management team followed the traditional Scrum process whereby the end product was to be developed incrementally through multiple iterations (i.e., Sprints). At the very beginning of this process, the project management team created a list of the product features that would be delivered (i.e., product backlog), which was based on the initial design and architectural documents produced by the Architecture and Development teams. As the project progressed from one iteration to the next, modifications were made to this product backlog and these modifications added, removed, and edited the original product features.

A two-hour planning session took place prior to each Sprint in order to define the Sprint's scope and to determine which project members needed to be involved, the allocation of these members (given as a percentage of the work time that each was to spend on the project), the tasks that these members had to perform and their duration, and the definitions of dependencies and/or deliverables. Members were also asked to report on any planned vacation or time away from work that they intend to take during the project.

Each Sprint lasted four weeks and had a limited scope, addressing one subset of the features identified in the product backlog. Through Stand-up meetings, the project members provided status

updates three times a week during each Sprint. At the beginning of the project when the team sizes were still manageable, the duration of each Stand-up meeting was 15 minutes. Later in the project, this duration was extended to 30 minutes.

During a Stand-up meeting, each project member gave a brief presentation about the tasks assigned to him or her and provided an update about which tasks were completed and which were still in progress. Members also reported how much time they had already spent on each task as well as provided an estimate of how long it would take them to complete unfinished tasks. Additionally, they identified the tasks on which they planned to work before the next meeting. If there were any roadblocks or dependencies that prevented them from proceeding with any of their tasks, they were encouraged to declare them as well as to raise any concerns and to discuss any potential risks.

With each status update, the project manager updated the Sprint file in order to reflect these status changes. At the end of each Stand-up meeting, the project manager generated a burn down chart that showed uncompleted tasks along with the remaining available hours for the project overall. This gave project members a good idea of the project's overall progress. Ideally, all of the tasks within a given Sprint were to be completed by the Sprint's end date; however, this was not always the case. There were multiple occasions when tasks from one Sprint had to be carried over into the next.

At the end of each Sprint, a 60–90-minute Sprint review session was held within which each team presented on what they had accomplished in the previous Sprint and what they planned on addressing in the upcoming Sprint. If any deliverables were available at the time of these sessions, teams had the opportunity to demonstrate them to the rest of the group and to answer any questions.

After this review meeting was completed, a one-hour retrospective meeting was also held. This allowed project members to comment on how the previous Sprint had gone and to discuss which practices had gone well and which had not. Members were also asked to provide feedback about why particular tasks were not completed and what could be done to resolve the issues that had caused these delays and to improve the process. The project manager helped to define the actions that could be taken to ensure that the process improvement suggestions made by project members would be achieved. The project manager also followed up on improvement actions that were defined during the previous retrospective meeting.

4.5 Process Related Observations

4.5.1 Process Flow

“Business people and developers must work together daily throughout the project.” [42]

The agile principle above that is part of the agile manifesto, summarizes one of the major dimensions of all agile methods, which is to bring together customers and all project members at the beginning of projects and to ensure that they work together, communicate, and are aware of any changes that may take place during the development process.

It was evident in this project that, from an administrative point of view, the project teams were involved and invited to various meetings. In reality, certain team members were added to particular Sprints only because the project managers saw that these members had to be there for the project to satisfy agile principles. These members took part in the formal development process and allocated a percentage of their time for the project but had no tasks to perform and did not even attend most of the Stand-up meetings since they had no tasks on which to report.

Team Member	Sprint Allocation (% of full time)	Total Available Capacity (Hours)	Total Planned Effort (Hours)
Member #1	100.00%	120	103
Member #2	75.00%	90	80
Member #3	75.00%	90	72
Member #4	30.00%	36	12
Member #5	5.00%	6	0
Member #6	75.00%	90	32
Member #7	10.00%	12	0
Member #8	20.00%	24	19
Member #9	10.00%	12	0
Member #10	75.00%	80	16
Member #11	10.00%	12	8
Member #12	20.00%	24	0
Member #13	10.00%	12	0

Table 3 - Allocation example 1

Team Member	Sprint Allocation (% of full time)	Total Available Capacity (Hours)	Total Planned Effort (Hours)
Member #1	100.00%	120	158
Member #2	70.00%	96.6	127
Member #3	75.00%	103.5	130
Member #4	15.00%	10.1	9
Member #5	5.00%	6.9	0
Member #6	50.00%	69	42
Member #7	5.00%	6.9	0
Member #8	25.00%	19	23
Member #9	5.00%	6.9	0
Member #10	0.00%	0	0
Member #11	8.00%	11.04	12
Member #12	25.00%	34.5	0
Member #13	20.00%	27.6	32
Member #14	80.00%	86.4	68
Member #15	75.00%	69	78
Member #16	3.50%	5	0
Member #17	15.00%	20.7	8
Member #18	25.00%	27	23
Member #19	75.00%	93.5	100

Table 4 - Allocation example 2

The two tables above provide examples from two different Sprints and demonstrate that particular members reported their specific numbers of hours for the project (total available capacity) but had no tasks to perform (total planned effort).

In Table 3, we can see that member #12 allocated 20 percent of his or her time for work on this project. This translated into a total of 24 available hours; however, this team member had no assigned tasks for the Sprint in question (for a total planned effort of zero). Similar results can be observed in Table 4, which shows that member #7 allocated five percent of his or her time for work on this project. This translated into 6.9 hours of available capacity, but zero hours of planned effort.

The previous two examples show that as per agile principles, various team members were formally assigned to the project. It looked like all the teams that should be working on this project from start to finish had a representative involved from the very start. However, in reality, some members had very minimal involvement (and sometimes none at all) with the project.

4.5.2 Tasks Definitions and Durations

When following agile practices, it is expected that the tasks that were defined during the planning sessions would remain unchanged. However, if a change was necessary, project managers would keep track of such changes so that they could be used for future estimations and planning. If, at the end of a Sprint, certain tasks were not completed or had been completed earlier than expected, this information was used as feedback for the next Sprint planning session. This gave the project management team a good idea of each team member's capacity, as well as when certain milestones could be expected to be reached.

If, for example, a team member believed that a document review task would take two hours but the task ended up taking five, the team member would know how to better assess the duration of that task the next time that a technical review task was added to the overall process. The problem here was that, because team members were allowed to extend or shrink task durations as they pleased, it was difficult to define the actual time that a certain task should take.

One major dimension that was noted with the development of this project was the ways in which tasks were defined and tracked throughout the project lifecycle. During Sprint planning sessions, team members had the option to define the tasks on which they were going to work in order to build a particular feature. Members were allowed to freely add tasks that they thought were necessary and without technical leaders questioning whether or not these tasks were sufficient to achieve the desired goals and whether or not the tasks that they had defined constituted the optimal way to achieve such goals.

In addition to this freedom, task durations were defined freely. After listing the number of tasks on which they were going to work during a particular Sprint, team members would go back and begin to

define how long each task would take. Again, these durations were never validated or questioned, and no attempts were made to optimize these durations.

During the actual Sprints, team members were allowed to add, remove, and edit tasks as well as change task durations without maintaining records of their historical values. In essence, this defied the reason for completing a Sprint planning session

Table 5 shows that, during a given Sprint, the team attended 16 Stand-up (STU) sessions. The numbers represent how many hours each project member had left in his or her allocated capacity. A positive number means that this member had excess time, while a negative number means that his or her task durations had exceeded the amount of time allocated for these tasks during this particular Sprint. It can be observed that for some members the numbers changed from one Stand-up meeting to the next (these changes are highlighted in yellow). This indicates the addition, removal, or modification of a particular task. It is also observed that some members were added to the Sprint after that Sprint started; while others were removed halfway through (these instances are highlighted in gray).

Name	STU 1	STU 2	STU 3	STU 4	STU 5	STU 6	STU 7	STU 8	STU 9	STU 10	STU 11	STU 12	STU 13	STU 14	STU 15	STU 16
Member #1	0.4	0.4	0.4	24.4	0.4	-3.6	-3.6	-3.6	-3.6	-3.6	-3.6	-3.6	-3.6	-3.6	-3.6	0.4
Member #2		76.8	76.8	10.8	10.8	-5.2	-5.2	-5.2	-13.2	-13.2	-13.2	-13.2	-13.2	-13.2	-13.2	-23.2
Member #3								5.6	5.6	5.6	5.6	5.6	5.6	5.6	5.6	^{5.} ₆
Member #4	6.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
Member #5	5.6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Member #5	5.6	-4.32	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4
Member #7	-0.9	-0.9	-0.9	-0.9	-0.9	-0.9	-0.9	-0.9	-0.9	-0.9	-0.9	-0.9	-0.9	-0.9	-0.9	-0.9
Member #8	5.6	-4.32	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4
Member #9	0.96	0.96	0.96	4.96	4.96	4.96	-1.6	-1.6	-1.6	-1.6	-1.6	-1.6	-1.6	-1.6	-1.6	-1.6
Member #10	5.6	5.6	5.6	5.6	5.6	5.6	5.6									
Member #11	0	0	0	0	0	0	0	-4	-4	-4	26	26	26	26	26	26
Member #12	-13.4	-17.9	-17.9	-17.9	-17.9	-0.5	-6.5	-6.5	-6.5	-6.5	-2.5	-2.5	-2.5	-2.5	-2.5	-2.5
Member #13	-0.8	-0.8	-0.8	-0.8	-0.8	-0.8	-0.8	-0.8	-0.8	-0.8	-0.8	-0.8	-0.8	-0.8	-0.8	-0.8
Member #14	5.6	5.6	5.6	5.6	5.6	-9.4	10.6	10.6	10.6	10.6	10.6	10.6	10.6	10.6	10.6	34.6

Member #15								11.2	11.2	11.2	11.2	11.2	11.2	11.2	11.2	11.2	11.2
Member #16	1.2	1.2	1.2	1.2	1.2	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6
Member #17	-3.6	-3.6	-3.6	-3.6	-3.6	-3.6	-3.6	-3.6	-3.6	-3.6	-3.6	2.4	2.4	2.4	2.4	2.4	2.4
Member #18	5.6	5.6	5.6	5.6	5.6	5.6	5.6	5.6	5.6	5.6	5.6	5.6	5.6	5.6	5.6	5.6	5.6
Member #19	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Member #20	5.1	4.58	4.58	4.58	4.58	4.58	4.58	4.58	4.58	4.58	4.58	4.58	4.58	4.58	4.58	4.58	2.58

Table 5 - Over allocation example

Team Member	Sprint Allocation (% of full time)	Total Available Capacity (Hours)	Total Planned Effort (Hours)
Member #1	85%	106.4	106

Table 6 - Member #1 allocation

By looking at specific data (Table 6), we can see that member #1 has an available capacity of 106.4 hours and a planned effort of 106 hours. Table 5 shows that in the first Stand-up meeting (STU 1) this member possessed an extra available capacity of 0.4 hours. Member #1 had six assigned tasks assigned with the following durations:

- Task # 1 – 20 Hours
- Task # 2 – 20 hours
- Task # 3 – 10 Hours
- Task # 4 – 8 Hours
- Task # 5 – 24 Hours
- Task # 6 – 24 Hours

In Stand-up #4, task #6 was removed, which caused the member's extra capacity to increase to 24.4 hours.

In Stand-up #5, task #6 was re-added, which caused extra capacity to drop back to 0.4 hours.

In Stand-up #6, a new task was added (Task #7) with a duration of four hours, which caused this member's extra capacity to drop to -3.6. This meant this member's planned effort exceeded the time allocated for this project.

In Stand-up #16, two task durations were changed. The duration of task #6 was dropped from 24 hours to 12 hours, and the duration of task #7 was increased from four hours to 12 hours. This caused the member's extra capacity to become 0.4 hours. Another thing to observe in table 5 is that some members joined the project part way through the Sprint (members #3 and #15), and that some members left the project during the Sprint (member #10).

Looking at the data this example demonstrates, it can be seen how project members were able to add, remove, and modify tasks from one Stand-up to the next during a particular Sprint. It can be seen that even the members themselves were freely added and removed from Sprints after they have already started. As mentioned above, Scrum practices dictate that the tasks defined within a given Sprint should not be altered. This obviously did not happen here.

There is a good reason to avoid alteration to planned tasks. In the Scrum method, the rate at which a team finishes its tasks is used to calculate what is called the team's "velocity." A team's velocity value is used in future Sprint planning sessions in order to determine how much work to assign. For example, if a team starts a Sprint with a planned effort of ten hours and can only complete eight of these ten hours by the end of the Sprint, during the next Sprint planning session, it will be noted that this team can handle only eight hours of work and more realistic planning can be done. Defining and tracking different team velocities assists in the creation of more realistic project timelines. It also assists project managers in estimating the timing of major milestones. Furthermore, team velocities can be used as benchmarks when planning other projects.

This is another example that shows how the agile practices were tailored by this organization to fit the existing structure with its existing processes. With members working on other projects and having to balance the work needed for this project with other tasks they had to complete, the members were often changing their tasks to meet these demands.

4.5.3 Over-Allocation of Team Members

At the Sprint planning sessions, team members were asked to define their allocations, which were represented as a percentage of the time that they would allocate to work on tasks related to the project. It was noted that some members were always over-allocated, where the sum of their task durations within a given Sprint exceeded 100 percent of their time (assuming 40hrs/week). This issue came up multiple times for more than one person, and it was never fully addressed.

For example, over-allocated team members never engaged in review sessions with their leaders to determine whether their task durations might be optimized or adjusted. Project leaders never suggested the addition of another resource to assist with extra workload. Over-allocations were simply noted by the project manager, and the manager's expectations for the given Sprint were unaffected.

Below is an example of one member's allocation over the course of ten Sprints. This allocation changed within the same Sprint from one Stand-up session to the next; for this reason, the numbers below represent average allocation per Sprint.

Sprint	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Hours	14.73	-21.066	-17.25	-37.27	-9.28	-13.2	-5.08	-0.6	0.8	-22.72

Table 7 - Member allocation over multiple Sprints

This member had an extra capacity of 14.3 hours in Sprint #1. As he or she moved from one Sprint to the next, however, the sum of this person's tasks was greater than the time allocated for each Sprint. In Sprint #2, for instance, the sum of this member's task durations was almost 21 hours more than was allocated.

This person was over-allocated in eight out of ten Sprints and this over-allocation ranged from 0.6 to 37.27 hours. This over-allocation resulted in unfinished tasks being carried over from one Sprint to the next.

4.5.4 Team Size

Initially, the project team was made up of 14 members; more members were added as the project moved from one Sprint to the next. Below is the number of project members for each Sprint.

Sprint	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Members Count	14	18	18	19	19	21	20	20	20	20

Table 8- Team size over multiple Sprints

Due to the large number of members, Stand-up meetings were extended from 15 minutes to 30 minutes, yet these meetings were still not long enough to address all of the updates on which project members wanted to report. Project members were forced to keep their updates short. Instead of addressing the tasks on which they were working by description, they began to refer to them by using the tasks' numbers on the Sprint sheet. This made it confusing for those project members who were listening over the phone and did not have access to the Sprint sheet. Certain project members mentioned that this made the purpose of the meeting obsolete as they did not know for which tasks the other members were giving updates.

4.5.5 Conflicts

Various conflicts arose during the course of the project. Most of these conflicts were minor and were easily and quickly addressed; however, one particular conflict created tension between a few of the teams and it took multiple weeks and numerous meetings to reach agreement about how to resolve it.

The architecture and design of the new internal product in development relied on the replacement of some existing components with new components that would deliver similar functionalities. One of the components that were going to be replaced was a database system upon which the Release Testing team relied for multiple reporting tasks.

Because the project involved an agile approach, the Release Testing team was involved in multiple meetings at the early stages of the project within which the architecture and design of the new components were discussed. It is difficult to say whether the Release Testing team was attending these meetings but did not pay close enough attention to what was being discussed, whether they never actually reviewed the architecture and design documents, whether these documents were not clear enough, or whether the replacement of the database was simply never discussed, but the Architecture team seemed to have assumed that the Release Testing team had no problems with the product's design and that the latter team had acknowledged and accepted the database replacement.

The conflict arose when the development cycle was almost done. A requirements gathering session was held in order to ensure that changes to the existing components were understood by the teams who would be impacted by these changes. In addition, the Development team also built an extra component that replaced most of the critical reporting functionalities that were provided by existing components. This meeting aimed to ensure that this replacement component is also understood by the rest of the teams.

At this point, the Release Testing team raised the concern that, although their database was being decommissioned, they still needed to perform their regular reporting tasks in the same way, and that, for this reason, the development team needed to build them a new component in order to provide the same functionalities.

The Architecture and Development teams were upset with this request because they were already behind schedule and were trying to finish their work. Also, they disagreed with the Release Testing team that the functionalities that they were asking for were in fact critical. They explained to the Release Testing team that the components of the new product would behave in a different fashion than the existing components and that this would cancel the need to accomplish the same reporting tasks.

The Release Testing team argued that if, for any reason, the new product's components did not deliver on what they had promised, the team would have to revert to performing its reporting tasks the way in which they do them today. However, since their present database system was going to be taken away, they would be stuck and would need an alternative.

Several meetings were held in an attempt to reach common ground; however, these meetings mostly ended in frustration and finger pointing. The Architecture and Development teams accused the Release Testing and Operations teams of not understanding how the new product would work and of being opposed to change. For their part, the Release Testing and Operations teams said that the Architecture and Development teams did not understand the importance of these reporting tasks because they were unable to see what actually happens when new products are integrated and deployed and that, when a critical issue does come up, the results of the reports that the Release testing and Operations teams generate are what makes resolving them possible.

In the end, all of the teams agreed that, for the time being at least, no further development would be done, but because the product would be introduced in multiple phases, new requirements would be considered before the phase in which the database system would in fact be removed.

4.5.6 Lessons Learned

The project management team held a “lessons learned” session after all development activities were completed and before the beginning of deployment. The purpose of this session was to review how the teams’ first Scrum project had gone as well as how it had been perceived by different teams. Feedback was gathered through a survey prior to this review meeting. Further feedback was provided during the meeting, and action items were discussed. Twenty-eight individuals accessed the survey, but only 13 fully completed it.

Items that were mentioned as helpful aspects of the Scrum process included:

- that it promoted good team work and better communication;
- that various teams liked the agile approach (e.g., demos, progression from product backlog to Sprint);
- that the development team was able to carry on work early in the project without having to wait for documentation; and
- that most of the members stayed on top of any issues that arose.

The items that were mentioned as unhelpful aspects of the Scrum process were:

- planning/estimating without complete information;
- use of a large team;
- resource constraints;
- deadlines for testing were not extended;
- the product backlog was task-oriented and not requirements- or use case-driven; and
- a lack of status meetings.

4.6 Summary

It can be observed that various factors affected the execution of this project. Introducing a new product to the company's existing system while at the same time changing completely the ways that the teams executed their projects (i.e., through the use of Scrum) was a big challenge and it changed the dynamics of the teams that were involved.

One major thing that can be observed is that the team members' roles were not redefined in any way to fit the new agile process. These members followed the traditional waterfall method when assigned to other projects. Then they were assigned to a new agile project, however, their roles in the new project were still the same as it would have been under the traditional waterfall process. In theory, they appeared to be engaged from the beginning of the project. Their actual involvement though took place only when the product was ready for them to work on it, and all of the previous project phases were completed (as in a typical waterfall process). As a result, some members became unaware of how the product development plan progressed and when it was time for them to take over the work, various conflicts and misunderstandings came up.

Another issue to point out is the over allocation of some members almost in every Sprint. This issue along with the poor management of tasks definitions and durations caused the project to go over time and over budget. The members seemed to struggle with their attempts to fit the newly introduced agile process. They were not instructed to change their traditional way of performing their tasks before they were shifted to follow the agile practices. It was up to the team members to adjust to the newly introduced agile process and somehow make things work.

The following sections present the results of interviews with different members of the project. These results are analyzed in conjunction with the observations described in this section so as to present a possible explanation of the observed results.

5. Interview Results

Twelve participants were interviewed individually and asked a series of questions (Appendix A). This was done in order to gain an understanding of their backgrounds and experiences within the company as well as within their current teams.

The interviews lasted between 20 and 50 minutes. Each participant was asked to briefly describe his or her day-to-day tasks and responsibilities along with his or her interaction with the project in question (the first project to be conducted using an agile method in the department). Participants were also asked to name those teams with which they interacted on a regular basis and to give examples of helpful and unhelpful behavioral patterns that the other teams demonstrated.

The interview questions aimed at exploring each team's perceptions of the other teams. The participants were encouraged to provide examples of their experiences working with the other teams. They were also asked about their perceptions of the agile methodology and Scrum in particular. These comments were aggregated, categorized and summarized in the upcoming sections.

Participants and responses were grouped together in high-level, general categories and placed in an order that is sometimes different from the order in which they were originally provided. For each "helpful" and "unhelpful" category, a number is added in brackets in order to indicate how many times this category was mentioned in the interviews. It should also be noted that each participant did not necessarily work with all of the other teams and for this reason might have had nothing to say about the teams with which they did not work.

5.1 Perception of Scrum

As stated earlier, the company in question adopted the Scrum method as its agile software development methodology. Interview participants were asked to describe their experiences working with Scrum practices for the first time in the context of this particular project. they were asked specifically to provide examples of how managing the project by using an agile method affected their daily tasks. Responses are categorized in Table 9

Helpful	Unhelpful
<ul style="list-style-type: none">• <u>Early Involvement (3)</u>• <u>Good Tracking of Project Status(5)</u>• <u>More Flexibility (2)</u>• <u>Early Deliverables/Demos(3)</u>• <u>Time Savings (1)</u>• <u>Early Discovery of Problems (1)</u>	<ul style="list-style-type: none">• <u>Extensive/Long Meetings (9)</u>• <u>Attendance Problems (2)</u>• <u>Poor Communication (2)</u>• <u>Lack of Collaboration (2)</u>• <u>Distorted Purpose of Meetings (2)</u>• <u>Process Problems (5)</u>• <u>Useless Deliverables (1)</u>• <u>Lack of Technical Leadership (2)</u>

Table 9 - Perception of Scrum

When asked about helpful aspects of the Scrum process, they indicated that the new methodology engaged different teams early enough in the project. They reported that this had a positive impact on how various tasks were conducted. The following is one participant's comment:

“Agile is good in the sense that the testers got involved in the requirements phase and got to work with the developers so that they were involved early on, and knew how to test the requirements as they come along.”

Participants also indicated how different tools and meetings used in the agile process (e.g., the Stand-up meeting, retrospectives) helped them to keep track of the project's status. The process also kept the communication between the teams very structured and formalized. Within each Sprint, every team

member knew exactly which meetings to attend and how to report on their tasks. This was done in a straightforward manner.

The Scrum process is formed from multiple Sprints, each with a limited scope or set of tasks, and certain team members understood this as one factor that assisted teams in becoming more flexible with their tasks. Work was done in small increments and any change could be easily spotted. Affected teams were able to react to any change that arose and adjust their tasks accordingly.

The Scrum process also allowed different teams to work concurrently, which had a great effect on time savings. This also allowed teams to discover problems and bugs early on and to fix them with minimal impact to the overall project.

When asked about non-helpful aspects of the Scrum process, a majority of the participants indicated that the number of meetings they had to attend was of great significance to their work. There was a lot of overhead associated with disconnecting from the tasks on which they were working before the meetings and then connecting back to these tasks after the meetings were completed. Additionally, the participants had to prepare for these meetings in order to be able to provide meaningful status reports. The number of meetings that participants had to attend - alongside meetings for other projects - took up a big chunk of their days and they viewed these meetings as a waste of time. The following is one participant's comment:

“Most of the people are taking this agile methodology as something that was slowing them down because they have to attend different meetings.”

Furthermore, it was difficult if not impossible to find a meeting time that worked for everyone. Team representatives were not always able to attend every meeting and provide status updates, and this resulted in incomplete views of the project's status.

Because of the limited duration of the Stand-up meetings, little communication flowed between the different teams and many discussions were taken offline, which deprived the rest of the group from being able to follow up on the resolutions or decisions made in these offline meetings.

Short Stand-up meeting duration also negatively affected the purpose of the meetings. As noted earlier, some team members tended to refer to their tasks by their numbers rather than by their descriptions, and others members who were on the phone had no idea to which tasks they were referring. The purpose of the Stand-up meetings was to spread awareness of different teams' progress, but these meetings ended up involving only vague status reporting sessions.

At the end of each Sprint, each team possessed a set of deliverable to demonstrate to other teams, and some of these demos were deemed "useless" by their owners because they knew that the Sprint's duration was too short to produce a meaningful deliverable. Because they had to deliver something, they sometimes delivered a "half baked" deliverable that they knew could not be tested. The following is one participant's comment:

"You can have deliverable at the end of each iteration but what you deliver you know it is lacking some functionalities so that it can't be tested. Testers will get back to you saying it doesn't have this feature, but you already know that."

Finally, the fact that team members were able to add, remove and defer tasks during any Sprint created confusion as to whether the addition or removal of these tasks was in fact essential and whether these additions and removals were the best approach for achieving the team's overall goals. In general, technical leadership, where team leaders or managers decide whether or not a given task should be added, was highly preferred by one of the participants over letting the project members themselves freely manage their own tasks.

5.2 Perceptions of the Architecture Team

The Architecture team is responsible for designing new products as well as adding new features to existing products. It is usually one of the first teams to be engaged when a new product is initiated. They were considered the product owner of this project and had the authority to decide which features to consider and which features to eliminate as well as how to prioritize these features.

Helpful	Unhelpful
<ul style="list-style-type: none">• <u>Taking Ownership (1)</u>• <u>Good Response Time (2)</u>• <u>Provides vision (3)</u>• <u>Provides Technical Assistance (4)</u>• <u>Good Communication (1)</u>• <u>Collaboration (3)</u>	<ul style="list-style-type: none">• <u>Lack of Involvement (1)</u>• <u>Lack of Understanding (5)</u>• <u>Disagreement (4)</u>

Table 10 - Perception of the Architecture team

Ten participants identified the Architecture team as a team with which they interact on a regular basis. These participants provided the following examples of helpful patterns of behavior.

The architects tend to take ownership of projects. They act as the product owners, and they review all of the features that are being developed and any requirements that are being captured in order to ensure that these features and requirements align with the overall vision that they have set for a given project. They also provide the members of other teams with necessary technical assistance and respond to their inquiries in a timely manner. They communicate clearly with other teams and collaborate with them in order to ensure that everyone stays on the same page. The following is an example of a participant's comment:

“They are the glue between different groups and facilitate communication.”

As for non-helpful patterns of behavior, interview participants stated that the architects are not as heavily involved in implementation as they should be. The architects tended to see things from a

particular point of view that sometimes lacks an understanding of the specific ways in which certain teams operate. They sometimes make assumptions about the feasibility of certain project tasks with regard to how soon or how easily these tasks can be completed that are sometimes far from the reality. They are also very opinionated and tend to discard others' requests when conflicts arise. They try to have their own way, even if it means disagreeing with others. The following are examples of participants' comments:

“Architects are sometimes superficial; they assume that things they design should work, though they are not sometimes as easy as they think.”

“They do not always understand how things work in reality.”

5.3 Perception of the Development Team

The Development team is responsible for translating the product's design and architecture into code. They develop code for new products as well as for new features that are introduced into existing products. The Development team ensures that these new features are integrated properly within the existing system. They also help with troubleshooting issues that may be related to code deficiencies.

Helpful	Unhelpful
<ul style="list-style-type: none">• <u>Good Response Time (3)</u>• <u>Takes Ownership (1)</u>• <u>Good Communication (4)</u>• <u>Provides Technical Assistance (5)</u>	<ul style="list-style-type: none">• <u>Poor Work Quality (2)</u>• <u>Bad SW Release Time (1)</u>• <u>Disagreement (1)</u>• <u>Poor Communication (2)</u>• <u>Lack of Ownership (1)</u>• <u>Co-location problems (Lack of Trust/Comfort) (3)</u>• <u>Availability (4)</u>• <u>Lack of Documentation (3)</u>• <u>Slow Response Time (1)</u>

Table 11 - Perception of the Development Team

Nine participants identified the Development team as a team with which they interact on a regular basis. They provided the following examples of helpful patterns of behavior that they have observed/experienced.

The developers always take ownership of the tasks that are assigned to them, and they respond to any inquiries in a reasonable timeframe. They communicate well with the other teams and help to clarify issues that are code-related in a simple manner. The following is one participant's comment:

"They give you the answers you need to do testing."

The participants also identified the following non-helpful patterns of behavior. Because the development team has to perform regular coding tasks in addition to providing support for production (end customer) issues, they were sometimes unable to manage the two effectively. The development team was involved in providing support efforts for production issues if it was suspected that their cause could be a problem with the code. It was perceived that the development team provides a poor support service. The following is one participant's comment:

"There should be a group that just supports production issues versus a group that works on development. I think we have failed to do that and as result there is no effective support provided by development team."

The Development team also kept pushing the release timelines for their software, which left the downstream teams with very little time to complete their tasks and meet the project's overall deadline. Also, when support issues are escalated to them they tend to push back. The communication between them and other teams at this point tends to be bad because they do not like to hear that their code may have caused problems in production.

Most of the time, developers are quite busy and this affects their availability and response time. This lack of time also affects the Development team's ability to provide initial documentation that must be referenced by other teams until official documentation is available.

Finally, the fact that the Development team is not co-located with most of the other teams creates a huge fracture in personal relationships; other team members do not have a sense of trust or comfort with members of the Development team.

5.4 Perception of the Functional Testing Team

The Functional Testing team is responsible for testing developed code. They also ensure that delivered functionalities match those that have been specified in design documents.

Helpful	Unhelpful
<ul style="list-style-type: none">• <u>Good Response Time (1)</u>• <u>Collaboration (2)</u>• <u>Good Work Quality (6)</u>• <u>Provides Technical Assistance (3)</u>	<ul style="list-style-type: none">• <u>Poor Reference to Documentation (1)</u>• <u>Lack of Technical Agreement/Understanding (3)</u>• <u>Poor Communication (1)</u>

Table 12 - Perception of the Functional Testing team

Seven participants identified the Functional Testing team as a team with which they interact on a regular basis. They have provided the following examples of helpful patterns of behavior that they have observed/experienced.

The Functional Testing team provides great technical assistance and always responds in a timely manner. They try as much as possible to collaborate with other teams in order to ensure that they understand what is being worked on. The quality of the work that the Functional Testing team delivers is excellent and they try to comprehensively test the software.

“If there is something wrong with the product, they can find it unless it is very subtle or a corner case that would not easily come up.”

Participants also identified the following non-helpful patterns of behavior.

The Functional Testing team tends to ask questions that can easily be found in existing documentation. They also provide big documentations that cannot be easily referenced when other members want to find out what is or is not being tested.

When production issues arise, the Functional Testing team tends to have a different understanding of the problem at hand than other teams like Operations and Release Testing, which causes the issue to take longer to be addressed or resolved. The same applies to identifying defects.

“Sometimes they do not have the understanding of the requirements or the design and if they do not know how it supposed to work, they cannot tell what a defect is and what is not.”

Finally, the Functional Testing team is regarded as isolated and as not having much interaction or communication with the members of other teams.

5.5 Perception of the Release Testing Team

The Release Testing team is responsible for testing how new products will be released and integrated with existing infrastructure. They are also responsible for tuning and configuring these products in order to provide the best possible performance.

Helpful	Unhelpful
<ul style="list-style-type: none">• <u>Good Response Time (2)</u>• <u>Approachable (1)</u>	<ul style="list-style-type: none">• <u>Lack of Ownership (2)</u>• <u>Late Changes (1)</u>• <u>Conservative (1)</u>• <u>Engagement Problems (8)</u>• <u>Poor Communication (2)</u>• <u>Lack of Extensive Research (4)</u>• <u>Poor Work Quality (6)</u>• <u>Low Technical Skills (2)</u>

Table 13 - Perception of the Release Testing team

Nine participants identified the Release Testing team as a team with which they interact on a regular basis. They provided the following examples of helpful patterns of behavior that they have observed/experienced.

For the most part, the engineers are approachable and respond in a timely manner.

“They provide reasonable feedback within the available time they can allocate for our tasks.”

Participants also identified the following non-helpful patterns of behavior. In general, the Release Testing team appears to be caught up in managing production issues and this seems to affect the quality of their work. They are viewed as very conservative when it comes to embracing change and new ideas. The fact that they are so busy makes it difficult for them to provide feedback in a timely manner; it also makes it difficult to engage them in technical conversations. At times when the Release Testing team was

engaged, they barely pay attention to what is being proposed and only respond when things come their way. They are also viewed as having low technical skills, which is a result of not researching issues ahead of time and ending up in situations within which they do not know what to do and in which they end up trying resolutions that may not be optimal. One of the participants mentioned that this is not an issue of not having enough time or resources; it is an issue of low technical skills.

“Is it a resource issue? I don’t think so, not in the sense that you do not have enough resources but rather it is a quality issue.”

5.6 Perception of the Operations Team

The Operations team is responsible for releasing new products and features. They decide the release times for products in every location that the company serves. They also assess the necessary resources for each release.

Helpful	Unhelpful
<ul style="list-style-type: none">• <u>Takes Ownership (2)</u>	<ul style="list-style-type: none">• <u>Poor Communication (3)</u>• <u>Slow Response Time (2)</u>

Table 14 - Perception of the Operations team

Three participants identified the operations team as a team with which they interact on a regular basis. They provided the following as examples of helpful patterns of behavior that they have observed/experienced.

The operations team members take responsibility for and ownership of issues that are not even necessarily their own. They do testing on behalf of the engineers when needed and never drop a task that is assigned to them.

“They do not drop any tasks.”

Participants also identified the following non-helpful patterns of behavior. They are a very busy team and it is difficult to get their attention. Their responses are also slow.

“They used to give us good feedback but not anymore. They used to read our docs but not so much recently.”

5.7 Perception of the Monitoring Team

The Monitoring team is responsible for monitoring released products in order to ensure that they work properly. This team is also responsible for resolving or escalating any production incidents.

Helpful	Unhelpful
<ul style="list-style-type: none">• <u>Takes Responsibility (1)</u>• <u>Good Response Time (1)</u>	<ul style="list-style-type: none">• <u>High Turnover (1)</u>• <u>Lack of Expertise (1)</u>• <u>Poor Judgment/Low Work Quality (1)</u>

Table 15 - Perception of the Monitoring team

One participant interacted with the Monitoring team on a regular basis and provided the following example of helpful patterns of behavior that he or she observed/experienced.

Because the Monitoring team is on the front line, it takes responsibility for any issue when it comes up and tries as much as possible to respond in a timely manner.

“They are the ones working day and night in shifts to keep an eye on the system.”

The participant also identified the following non-helpful patterns of behavior. Due to the nature of the job (e.g., having to rotate through different shifts, dealing with stressful situations on a regular basis, not enough pay), the Monitoring team appears to struggle with keeping their resources and tends to have a high turnover rate. This impacts the quality of the work that they produce. Experienced members leave the team and new members must struggle until they get up to speed and gain a better understanding of the details of their day-to-day tasks.

“They escalate issues that might not need escalation and miss important ones.”

5.8 Perception of the Networking Team

The Networking team is responsible for making any change to the company's network in order to support the release of new products and features.

Helpful	Not Helpful
<ul style="list-style-type: none">• <u>Good Response Time (1)</u>	<ul style="list-style-type: none">• <u>Poor Communication (1)</u>• <u>Slow Response Time (1)</u>• <u>Lack of Professionalism (2)</u>• <u>Poor Work Quality (1)</u>

Table 16 - Perception of the Networking team

Two participants identified the networking team as a team with which they interact on a regular basis. They provided the following example of helpful patterns of behavior that they observed/experienced.

When dealing with the Networking team within the realm of testing and staging, the team provides responses in a timely manner.

“We have a very positive experience with them in staging; they tend to get things done relatively quickly.”

The participants also identified the following non-helpful patterns of behavior. The perception of the participants seems to be reversed when asked about how the networking team deals with production issues. While the networking team seems to be very responsive when it comes to the development, testing, and staging issues, it tends to exhibit slow response times and avoids in-person conversations when it comes to supporting production issues. This in turn affects the communication between the networking team and other teams. The Networking team also reflects an unprofessional image when dealing with production issues because its first response is to push back and deny responsibility when these issues arise. Indeed, they must be “forced” to investigate further.

“They avoid taking responsibility.”

5.9 Perception of the Project Management Team

The Project Management team is responsible for managing various projects in order to ensure that every team provides the required deliverables within specified times and budgets.

Helpful	Unhelpful
<ul style="list-style-type: none">• <u>High Level of Professionalism/Good Work Quality (8)</u>• <u>Good Planning/Management(3)</u>• <u>Good Communication (2)</u>	<ul style="list-style-type: none">• <u>Poor Management Skills (7)</u>• <u>Poor Project Status Delivery (3)</u>• <u>Forces Schedules (1)</u>• <u>Extensive Meetings (2)</u>• <u>Poor Communication (4)</u>

Table 17 - Perception of the Project Management team

Nine participants identified the project management team as a team with which they interact on a regular basis. They have provided the following examples of helpful patterns of behavior that they observed/experienced.

Participants saw a lot of value in the Project Management team taking over responsibility for organizing meetings and keeping track of different tasks and their progress. Participants also believed that the level of communication provided by this team was good enough to keep them up to date about the project's overall progress and status.

“In terms of estimating and planning, I think they have done a good job just relying on what we provide rather than forcing their schedule.”

Participants also identified the following non-helpful patterns of behavior. On some occasions, the Project Management team loses control over the meetings and the meetings' participants begin to digress. This is regarded as a huge waste of time because further meetings must then be scheduled in order to address issues that were not fully discussed in the first meeting.

Furthermore, some participants struggled to gain a clear understanding of the different tasks of the project and how these tasks were progressing. They believed that the communication provided by the Project Management team lacked detail and that they were forced to reach out to other resources in order to get the information that they needed.

A couple of issues were raised that might not be wholly related to project management but were still regarded as the responsibility of the Project Management team. One of these issues was the forcing of certain schedules or timelines onto project participants.

“There might be an emergency where they tell us you only have two days and have no choice.”

The other issue involved booking a large number of meetings. This is mainly a result of following an agile methodology.

“Too many meetings and scheduling conflicts.”

5.10 Perception of the Requirements Team

The Requirements team is responsible for gathering product and feature requirements from different project stakeholders. The requirements that they gather act as a contract that specifies which features will be delivered at the end of the project as well as the changes that will be needed to accommodate these changes.

Helpful	Unhelpful
<ul style="list-style-type: none">• <u>Cross-Team Communication (1)</u>• <u>Plays an Important Role (1)</u>• <u>Good Work Quality (1)</u>• <u>Good Requirements Gathering Process (2)</u>	<ul style="list-style-type: none">• <u>Requirements Review and Approval Problems (1)</u>• <u>Long Requirements Completion Time (1)</u>• <u>Poor Follow Up (2)</u>• <u>Low Technical Skills (2)</u>• <u>Unclear/Complex Requirements Wording (3)</u>• <u>Poor Communication (2)</u>

Table 18 - Perception of the Requirements Management team

Six participants identified the requirements team as a team with which they interact with on a regular basis. They provided the following as examples of helpful patterns of behavior that they observed/experienced.

The Requirements team does a good job of bringing different teams together and manages cross-team communication between these teams. They follow a well-structured process that helps in gathering requirements properly and in a timely manner.

“They handle cross-team communication and ensure that when there is a dependency or a conflict that the appropriate needed individuals talk to one another.”

The participants also identified the following non-helpful patterns of behavior. The Requirements team takes a long time to complete its requirements. This is regarded as impacting productivity because other teams must wait for them to do complete the requirement. The Requirements team is also not on top

of the projects in the same way as the project managers are, and when changes happen downstream, requirements are not necessarily updated to reflect these changes.

The Requirements team is also not a technical group, so others find it annoying to have to explain things to them in order to bring them up to speed on the projects.

“They have very low or completely no technical knowledge of things they are asking questions about. “

The team also words requirements in a specific format that may not always reflect what participants would like to convey, and the level of communication that they provide to other teams is not always up to those teams’ expectations.

5.11 Perception of the Documentation Team

The Documentation team is responsible for documenting product code, functionalities, and configuration settings.

Helpful	Unhelpful
<ul style="list-style-type: none">• <u>Useful guides for completed projects (1)</u>	<ul style="list-style-type: none">• <u>Lack of Documentation for Certain Projects (1)</u>• <u>Lack of Documentation Tasks in Sprints (1)</u>

Table 19 - Perception of the Documentation team

Two participants identified the documentation team as a team with which they interact on a regular basis. They provided the following as examples of helpful patterns of behavior that they observed/experienced.

The Documentation team usually provides good guides for completed projects. Most of the downstream teams use these guides when issues arise and these teams consider the guides to be very helpful.

The participants also identified the following non-helpful patterns of behavior. While a project is still in progress, there is usually not enough documentation to help the various teams understand how to carry on certain tasks. For example, when a test environment must be set up, teams usually refer to project's documentation in order to understand the components that need to be deployed. These teams use the Documentation team to determine whether there are any dependencies and how to configure certain parameters, among other things. A lack of documentation in such stages causes these other teams to spend more time investigating these issues. Also, when tasks are added to Sprints, the documentation team does not seem to add enough tasks to cover the needed work.

“There is not enough documentation.”

6. General Discussion and Implications

In an ideal world in which agile rules are easily applied to any company's processes, one would expect agile project teams to possess a number of important characteristics. Agile teams are expected to be responsive, to communicate effectively and easily, and to always be aware of each other's progress and project updates. It is also expected that any issues that can impact the project are discovered early and are resolved with minimal conflict and disruption. The project manager should be able to track the progress of individual team members and provide good estimations of when defined milestones and project completion will be reached.

In reality, this is rarely the case for most of large companies in the software development industry. The adoption and implementation of agile methods sometimes poses a real challenge. This was observed in this case study both through the feedback provided by participants and through the patterns and events that took place during the duration of the research.

This section introduces the limitations of this study. This will be followed by brief summaries of the major findings of both the observations and interview results sections. These summaries will assist in an attempt to explain the situation at hand and to identify various observed patterns and behaviors.

6.1 Limitations

This study possesses several limitations. The first limitation is related to the research methodology and design used in this study. The study relied on a qualitative approach, which, unlike quantitative methods, does not build on existing hypotheses but rather attempts to approach the research topic in an exploratory manner. The researcher who employs qualitative methods, tries to build an understanding of the situation at hand with minimal and sometimes no background of what is occurring within it. The researcher then extends further and provides his or her own explanation and analysis of the situation based on his or her personal perceptions. It is therefore obvious that this form of research depends heavily on the researcher's interpretations of what has been said and observed and that its quality is a reflection of the researcher's skill and diligence. For this reason, it is possible that the findings in this study may be based on incorrect interpretations of the data. It is also worth noting that I am an experienced professional who has worked on multiple agile projects prior to this study and possesses a good knowledge of both the research area and the study's participants. This has hopefully allowed me to correctly interpret, understand, and analyze my findings. Furthermore, assuming any bias existed as a result of me being a member of the observed project, the absence of prior hypotheses and assumptions at the beginning of the study worked against the existence of any bias.

The second limitation is related to the size of the sample. Its results are based on the responses of a limited number of participants. The comments and feedback provided by this group represent a small subset of professionals working in the software development industry and may not be representative of the industry as a whole. However, the variety of specializations and occupations of the professionals involved in this study make up for the small sample size and provide a good sense of the impact of the first-time adoption of agile methods within a variety of teams that possess different functional scopes.

The third limitation is related to the fact that the study observed only one project. This introduces the possibility that some of the observed patterns may be specific to this particular project and hence cannot be used to draw general conclusions about the adoption of agile practices. There is also a concern related to the scope of the project in question because the observed project introduced a new system to the organization. It is possible that the uncertainty associated with this project's scope may have amplified the uncertainty associated with the introduction of new agile practices. It may be that, if a project with a smaller and more traditional scope was introduced, different group dynamics would have been observed.

The last limitation is related to the time constraints of the study participants. The data represented in this study relied heavily on the feedback and comments provided by participants and captured in interviews. There were occasions when the study's participants could only dedicate a small amount of time to attend these interviews as they had many prior commitments and engagements that made it hard to allocate time to assist with this study. Had there been more time, more details and comments would have been provided and more patterns would have been explored and analyzed.

6.2 Observations Summary

The observations of this study were reported in a previous section. Below is a summary of the main findings to which these observations led.

This study examined a pilot project that introduced agile software development practices to a department for the first time. The teams involved in this project belonged to different departments within the company. Despite the new development process, activities were carried on a sequential manner that made it look as if the project teams were arranged in a waterfall-like structure that was based on their functional roles. This structure triggered different perceptions among the project teams. These perceptions depended upon where their teams were positioned. The upstream teams felt responsible for making most if not all of the decisions that were necessary to move the project forward. At times, they appeared to ignore the requests of the downstream teams. The downstream teams had the general perception that they had no say in how the project was developed and in the major decisions that were made about the project as it progressed.

From a process perspective, the agile practices that were undertaken by the project teams did not fully follow the recommended and defined agile practices. Agile methods experts specify that project teams should work together closely and in parallel from the beginning of a project. In the project under observation, it was seen that, from an administrative point of view, the project manager tried to pull all of the resources together from the start. It was also noted that although some members allocated percentages of their time to the project, they had no tasks assigned to them and often did not attend the Stand-up meetings.

Furthermore, the teams made modifications to task durations and definitions. Project members freely changed the durations of the tasks on which they worked within a given Sprint, and did so after the Sprint was initiated. They also had the liberty of adding and removing tasks as they pleased. This posed quite a challenge to the project managers, who had to keep track of the amount of work completed and the amount that remained. It also made it hard for the project managers to assess teams' abilities to finish particular amounts of work per Sprint (i.e., velocity) or to accurately estimate the time necessary to complete certain tasks or features. Project managers relied on asking members for their own estimations

of when these tasks or features would be completed. A related challenge was ensuring that, with such rapid changes, all project members remain informed and up to date on the project's overall status.

Over-allocation was another major issue. There are many different explanations as to why over-allocation occurred. However, it was interesting that this over-allocation was acknowledged by the project members but never fully resolved. Whenever a member has allocated 100 percent of his or her time to a project and is not able to deliver all of the tasks assigned to him or her, a solution should have been provided. This over-allocation may be the result of having a single, highly specialized resource who is the only person capable of delivering particular tasks. Nevertheless, technical leaders should have provided guidance in order to either break down the assigned tasks differently over the span of multiple Sprints or to assign more resources in order to address this workload. This issue caused delays in delivery timelines.

Agile methods experts recommend that project team size be kept small. The recommended size is seven, plus or minus two persons. For this project, team size started at 14 members and grew over time. This expansion in team size had an impact on the communication flow between the project's different teams. Stand-up meetings were the primary source of communication between project members. Members met three times per week for duration of 15 minutes in order to provide status updates to the project manager and other project members and to learn about others' updates as well.

Given this large team size, the 15 minutes that were allocated for Stand-up meetings were not enough for members to provide meaningful updates. For this reason, the meetings were extended to 30 minutes. Even with the longer meetings, however, it remained a challenge for the different members to provide all of their updates. Because of time constraints, project members began to refer to their tasks by their numbers on the Sprint sheet rather than by their descriptions so as to be able to report more updates in shorter durations. However, this made the updates confusing for some members, especially those who had no access to the Sprint file during the meeting. These members could not understand which tasks were being discussed, and hence did not fully benefit from the updates.

6.3 Summary of Interviews

One-on-one interviews were conducted with some project members. This section summarizes the main findings that were provided by these interviews.

Project members were asked to provide examples of other teams' practices and behavioral patterns that they perceived as either helpful or unhelpful in the context of collaboration on this project. Participants were also asked to provide feedback about the Scrum method and to list those practices that they found helpful and unhelpful. Below is a summary of the numbers of examples/comments provided by various members on each team.

It is worth mentioning that every project member did not necessarily comment on all of the other teams, but rather discussed those teams with which they often interacted. Also, the values in the "Overall Perception" column were created by determining the number of helpful and unhelpful examples that participants provided for each team. If the number of helpful examples exceeded the number of unhelpful ones, the perception of the team in question was considered positive and vice versa.

Area/Team	Total Number of Helpful Examples/Comments	Total Number of Unhelpful Examples/Comments	Overall Perception
Scrum	15	25	Negative
Architecture	14	10	Positive
Development	13	18	Negative
Functional Testing	12	5	Positive
Release Testing	3	26	Negative
Operations	2	5	Negative
Monitoring	2	3	Negative
Networking	1	5	Negative
Project Management	13	17	Negative
Requirements	5	11	Negative
Documentation	1	2	Negative

Table 20 - Overall perception of Scrum practices and each team

Table 20 demonstrates that there are in general negative perceptions of each team when using the Scrum method. The twelve participants also provided feedback about their experiences working with Scrum. They provided 13 distinct helpful examples and 23 distinct unhelpful ones, which indicates that their overall experience with these new agile practices was not positive as well.

Examples of positive feedback include:

“It is a very good tool for managers, team leads, and project managers to see how the project is going.”

“It is very good in tracking time and resources and seeing the progress and where is the strong side and where is the weakness in your team.”

A number of negative comments and examples were also provided. Below is an example of these comments:

“It is PM methodology, so a lot of the team members who are not familiar with it might find it confusing. It is a definite change of mindset around here, it is not something that we typically do, so that in some areas was challenging”

Individual teams also provided their perceptions of each other in the context of this collaboration. The majority of these perceptions were negative, as is shown above. It can be argued that these perceptions existed prior to the introduction of Scrum. However, these perceptions are still worth noting.

By taking a closer look at the results of the interviews, it can be noted that the category of communication was mentioned in nine out of the eleven teams examined. The table below shows the overall perception of the communication quality for each team. The values of the “Overall Perception” column were determined from the numbers of helpful and unhelpful examples provided for each team. If the number of helpful examples exceeded the number of unhelpful ones, the perception of the team in question was considered positive and vice versa.

Area/Team	Number of Good Communication Examples	Number of Bad Communication Examples	Overall Perception
Scrum	0	2	Negative
Architecture	1	0	Positive
Development	4	2	Positive
Functional Testing	0	1	Negative
Release Testing	0	2	Negative
Operations	0	3	Negative
Monitoring	0	0	N/A
Networking	0	1	Negative
Project Management	2	4	Negative
Requirements	0	2	Negative
Documentation	0	0	N/A

Table 21 - Perception of communication

Table 21 demonstrates that there was in general a negative perception of communication among the different teams. It was also perceived that agile practices did not provide enough good communication to satisfy the various project members' expectations.

“The fact that we had too many people interested in attending the Stand-up meetings meant that people were interested in knowing what is going on and they were not receiving this information any other way.”

Project members also indicated that communication did not flow smoothly. They felt that they did not always have a good understanding of each other's statuses. They also felt that they did not have a good means for communicating their own statuses effectively.

“The project manager did not provide good communication. They did not understand the why behind certain changes

6.4 Discussion

6.4.1 Communication:

Continuous, clear and effective communication is a key factor in the use of agile methods. These methods rely heavily on keeping all project stakeholders informed so that any obstacles or issues that occur are observed early and addressed in a timely manner. This is accomplished through the introduction of regular Stand-up meetings. These meetings ensure that project members are kept up to date on the status of each task and deliverable.

The co-location of team members with different technical specialties is a major factor on which agile practices rely because it contributes to effective communication between project members. Any questions that arise are immediately answered instead of occurring over phone or email. Also, co-location allows team members to exchange ideas and to provide assistance easily. Having project members work together in the same physical location builds a sense of trust between them. This allows project members to communicate openly and to share ideas.

In this study, it was observed that communication was not well perceived by the project's members. This was clear in both the interview results as well as in the general observations during the research period. Whether this negative perception should be attributed to the adoption of new agile practices or to existing cross-team dynamics, some explanations can be provided.

In this study, the project members were not fully co-located. The teams belonged to different departments that were located in other buildings in the same city or in different cities entirely. Project members relied mainly on email to communicate; if needed, conference calls were also organized. Some of the project members never actually met in person. Below are a few comments provided by one of the project members during his or her interview:

“When it comes to other members of the project, you are not working together every day so you do not have this sense of trust or comfort to be interacting with the person in a higher level.”

“Agile co-location is not necessarily required but it is better for an agile team.”

“There is a huge fracture from the personal relationship side among the project members.”

The comment below was provided by another team member:

“No one trusted anyone else to tell them what was going on. That is my own interpretation. That lack of trust does not mean they do not think that others will tell them stuff, it is more that they do not feel that whoever is in these meetings understands their needs well enough to represent them.”

From the comments above, it is clear that project members felt that there was a communication gap. Some members attributed this gap to the lack of personal relationships between project members. Co-location would indeed have helped reduce, if not fully resolve, the impacts of negative communication. This is why the agile methodology experts highly recommend co-locating project members from the beginning of any project.

Although, agile principles rely on co-location as a requirement for effective collaboration, in reality—and especially in big companies—it is not always possible to fully control co-location. As the number of employees within a company grows, it is inevitable that these employees will be scattered across different buildings or across different geographical locations. It is not feasible that, for each new project that is initiated, project members are moved in order to co-locate. This is also not possible when a resource is assigned to multiple projects. Limitations related to logistics, cost and resourcing highly impact the feasibility of co-locating the members of a given project.

Team size is another possible factor that may have impacted the quality of communication between project members. For agile methods, the recommended team size is between five and nine members. The observed project team began with 14 members and became 20 members by its end. Having a large team introduces a number of challenges to both the project manager as well as to project members.

The use of Stand-up meetings was one of the practices that was most impacted by the team’s size. The duration of these meetings had to be extended in order to accommodate the number of team members. However, extending these meetings did not help to provide clearer communication to project

members. As explained before, the new duration of 30 minutes was still not long enough for members to provide meaningful status updates. Additionally, there was not enough time for members to ask questions in order to verify or confirm their understanding of certain topics. Many of these questions had to be taken offline and discussed in separate meetings.

This practice created further problems. First, project members who were unable to attend the offline conversations were deprived from staying up to date on certain areas of the project. Second, the new meetings that were set up in order to address these inquiries consumed even more of the project members' time. This was perceived by some of members as unnecessary overhead that could have been avoided if better communication had been in place.

These consequences of having a large team instantiate why agile methods recommend a small team sizes. However, in reality, a project with a wide scope may require the involvement of various groups, which in turn translates into a bigger team. This is not a factor that companies can easily control or limit.

The last factor that may be a cause of the observed poor communication is the allocation and availability of resources. For each project to which a member is assigned, he or she is expected to provide allocation. This translates into a percentage of time that the member will spend working on a given project.

In theory, agile methodology assumes that a resource will be fully dedicated to a particular project at a given time. However, in reality, each project member may be working on a number of different projects at the same time. Each of these projects has its own set of meetings that the member must attend. Each project also has its own set of deliverables.

With this in mind, it can be understood how an individual project member's work day is occupied by multiple meetings. For each meeting, the member must disconnect from their work and then reconnect with their previous task when the meeting is finished. The time that is taken to disconnect from and reconnect to these tasks was perceived as a huge overhead by the project members, given the number of meetings they had to attend daily. Members felt that this time could have been put to better use and that, after attending these daily meetings, they were left with less time to finish their actual tasks.

This helps to explain why some project members tended to skip Stand-up meeting calls. This deprived them of receiving other project members' updates. They also had to provide their own offline updates to the project manager, which in turn meant that other project members did not receive these status updates.

Furthermore, some members would join Stand-up meeting calls only so that it was noted that they had attended. They would work on their own tasks during the call and pay little attention to what was being discussed. During these meetings, other team members would bring up concerns or risks that they felt should be shared with the entire team. Given that these members were not always paying attention to what was being said, it was assumed that they had no comments on the topic as well as that they had been sufficiently informed. If problems arose later on, these members were held responsible for being informed about these risks and not taking actions to address them.

The interview results also show that various negative comments were provided regarding other categories such as availability, response time, disagreement, and lack of ownership collaboration involvement. These comments can be considered as symptoms of a mismatch between the organizations current processes and agile requirements. Given that these teams had worked together on multiple previous projects, it might be expected that they had previously established stable processes. The feedback provided indicated otherwise. It was evident that most teams had issues communicating with one another. These issues were sometimes due to the unavailability of a certain team. At other times, this poor communication was a result of not being able to reach agreement with the other teams on specific matters.

After examining the factors above that could possibly be the causes of poor communication between the project members, it became evident that this company's organizational structure was not providing a healthy environment within which the teams could be optimally productive. The introduction of a new software development method was expected to be the magic wand that would fix all of these broken processes, but it was not enough. The existing organizational structure prevented these processes from becoming stable. It did not evolve and change to accommodate the adopted agile practices Furthermore, the introduction of agile practices created problems that had not existed before.

The organizational structure could have been altered in various ways to minimize –if not fully eliminate- the problems that have been already noted. Restructuring the current teams and redefining the roles of their members could have possibly changed the perceptions of the different teams and enhanced their experience working with agile practices.

The lack of trust between team members was attributed to the lack of co-location, however, if these members were part of the same department, this feeling of mistrust could have been minimized if not fully eliminated. Members of the same department or team feel that they share the same goals and objectives. Whether they are located physically in the same location or not, they still feel some sense of unity. Restructuring the teams prior to introducing the agile practices could have possibly assisted in bringing the different members of the project together. It could have also eliminated the conflicts and misunderstandings that took place during the development of this project.

The project members also struggled with adjusting their day-to-day tasks to follow the newly introduced agile practices. The team members were never instructed to alter the way they performed their tasks. No guidance was provided either on how to adjust to the new agile practices. They were left on their own to manage their daily work load and find ways to fit the new process. The management team in the organizations could have introduced some structural changes to ensure their team members assigned to the project were managed more efficiently.

The team members were overwhelmed by the number of meetings they had to attend for every project. They felt that there was a lot of overhead associated with attending these meetings, and being closely involved with the project. This led some of them to follow the traditional waterfall process in reality – as they have always done- while pretending to follow the agile practices on paper to make it look like they have adjusted to the new system. This disconnected those members from the rest of the project team, and accordingly caused the observed poor communication. The members struggle should have been noted and addressed, especially that various symptoms of their struggle could be observed throughout the project duration. The over allocation of some members, and the absence of others from the Stand-up meeting are a couple of examples of these symptoms

6.4.2 The Adoption of Agile Methods

In the software development industry, new technologies emerge almost every day. These new technologies trigger changes in companies' processes. The end customer's expectations are constantly changing as well, and this adds to the changes that companies must address.

The adoption of a new methodology is an example such a change. When a company decides to shift to using agile methods, there is an unavoidable impact to its structure and culture. These changes in turn have a great effect on how the adoption process is both perceived and implemented. Various examples of the successful adoption of agile methods can be observed. The same applies to unsuccessful attempts. This leads us to wonder about the factors that support successful adoption as well as those that represent major obstacles to adoption.

A number of factors can be highlighted as reasons why the adoption process was not as successful as expected in the environment in which this study took place. Below is a summary of these factors along with interpretations of the reasons behind each one.

The first factor is related to the poor administration of the project teams. The members felt that the agile practices were forced on them regardless of whether or not the practices fit what their daily work. They did not feel that they were well prepared for this shift by their management team. They also felt that they did not receive enough guidance or support from their superiors to help them adjust to the new practices. A lack of understanding of the different members' roles could be observed. The project members were expected to follow the agile practices without assessing the fit of these practices to the members' daily activities.

The overhead of multiple Stand-up meetings is an example of an area where the project members struggled. The comment below was provided by one of the project members during an interview.

"All of these Stand-ups that are so annoying. You have to stop working to attend them then go back to your work after. With them being in the middle of the day like that it interrupts what we do. This time for most of us is the time where we get things done."

Prior to the introduction of agile practices; the team reported their statuses through a different channel. With the introduction of an agile method, Stand-up meetings became the official channel through which all of the project members came together and updated one another. With project members being assigned to more than one project at a time, attending Stand-up meetings consumed big portions of their time.

It's also interesting to note that the reporting structure within this project was very similar to how reporting is done in a Matrix Departmentalization structure [2]. The Matrix Departmentalization or Matrix Organization structure is an organizational structure that simply places an employee on a grid in order to represent his or her reporting hierarchy.

Within this grid, the functional line and the product or project line are overlaid in order to form a matrix. An employee will belong to two groups at the same time, a functional group and a product or project group. He or she will also have two managers to whom to report. His or her permanent manager is the manager in the functional area and is reported to vertically. Horizontally, the employee reports to a second manager on either a temporary or permanent basis. For example, he or she might report to a project manager on a temporary basis and change project managers each time that the employee is assigned to a new project. Alternatively, employees might report to a product manager on a permanent basis in a matrix structure defined in terms of both functional and product groupings.

Prior to adopting agile methods, project members would report their progress and status updates to their functional manager only. This was done on either a weekly or bi-weekly basis via email or in-person. After adopting agile, project members found themselves reporting status updates and progress twice. They continued to report to their functional managers, but they also started reporting to the project manager for each project to which they were assigned. The frequency of reports to the project manager was also high given that project members had to attend Stand-up meetings three times a week. This reporting structure increased administrative and managerial overhead. It also added reporting tasks to the workloads of the project members, and this in turn negatively affected their perceptions of agile practices.

Prior assessment of the end-to-end development process where each area in the process is examined for possible enhancements should have been considered. Forcing the agile practices on the project members led to a general sense of frustration. Project members should have been consulted before

the introduction of these new practices. These practices should have also been tailored to best fit what they accomplish on a daily basis along with a redefinition of the roles and duties of the project members.

Another factor that may have affected the ability of the company to easily adopt agile practices was the choice of the product. In this case study, the company adopted agile practices for the first time in order to develop a new product. The product in question was a new idea that introduced major changes to the existing system. A lot of the components of this system had to be reexamined in order to assess how they would work with the new product. There was a great deal of ambiguity regarding the behavior of the overall system within the new scenarios that this new product introduced.

Project teams had to go through a learning curve in order to understand the expected behavior of the new product and to identify what needed to be changed in order to accommodate its new functionalities. This was not an easy task given that they had to work on other projects at the same time as well as address other high priority issues. Meanwhile, they faced another learning curve. They had to familiarize themselves with agile practices and alter their daily tasks in order to fit the new agile development model.

The ambiguity introduced by the new product was amplified by the ambiguity introduced by the adoption of the new agile practices. Project members felt that they had to learn a great deal in many different areas and in very little time. They also had to ensure that they did not impact other projects and tasks on which they were working during the process of learning both the new product and the new agile practices. This was a huge challenge.

Given that the new product idea was still being formulated, its requirements were changing constantly. Nevertheless, project members were expected to continue working off of the latest requirements draft and to adapt to whatever changes were introduced. Given that teams were used to having a clear and final set of requirements defined before they began their work, this was not an easy task. They struggled to understand the agile practices and to alter the ways in which they performed their tasks. At the same time, they struggled to understand the technical aspects of the product itself in order to define what they needed to deliver.

In this particular case, it may have been a better choice to introduce agile methods within a different project scope. Developing new functionalities for an existing product would have been a better fit. This would have allowed project members to adapt to the new development methodology more easily.

7. Conclusion

This exploratory study examined the challenges and limitations that surface when agile software development methodologies are adopted for the first time within an organization. This study focused on observing the interactions between the different project members throughout the duration of a given project. It also relied on interviewing the project members to gather further details about their perception of other teams as well as the agile practices.

The study observations and interview results indicated that the observed organization was not fully successful in adopting agile methodologies. It can be easily seen that this organization introduced the agile practices without any prior assessment of their current processes. They assumed that the agile practices were enough to ensure a successful delivery of their new product. In reality, these practices were not enough to guarantee a timely and proper product delivery.

The main findings of this study lead us to conclude that the structure of the observed organization was the main limiting factor, causing the adoption attempt to not be as successful as expected. This existing rigid structure limited the ability of the organization to evolve and change. The existing structure was consistent with the Waterfall software development process and was left unchanged when the agile process was adopted. This led to various adoption problems that were observed during the project.

Communication is one of the main pillars that agile practices rely on. Effective and timely communication ensures that the agile practices deliver the benefits they promise. In this study, it was observed that the quality of communication between different project members was perceived as being very poor. The lack of co-location, large team size, allocation and availability of team members were factors that triggered this negative perception of communication among the project teams. Other categories such as availability, response time, disagreement, and lack of ownership collaboration involvement were also negatively perceived.

Another factor to mention is the adoption process of the agile practices which was not well introduced. This could be observed through the struggles that each of the project members had to go

through on a daily basis to adjust their regular activities to the new practices. The management team introduced the agile practices to the project members without providing enough guidance and support. The project members were frustrated with the fact that they had to follow some practices that provided little or no benefit to them. They ended up finding ways to work around the existing system, which led to failure in fully benefiting from the introduced agile method.

Also selecting a completely brand new product idea to develop in conjunction with the introduction of the agile practices for the first time to the organization was not an optimal choice. The ambiguity introduced by trying to figure out the details of the new product was amplified by the ambiguity introduced by the adoption of the new agile practices. This is another factor that caused the project members to struggle with this particular project. They felt pressured from a technical perspective to understand the new product and assess the impact of introducing it on their existing systems. At the same time, they were presented with a new set of practices to follow that they also had to adjust to as quickly as possible.

Examining each of these factors closely leads us to conclude that the existing organizational structure was the root cause of the observed problems. The structure of the organization seemed to have this waterfall like flow from an operational point of view. When agile practices were introduced, no alterations or changes of any kind were done to adjust to it. The management team assumed that the practices were easy to follow and that the project members should have no problem in adapting to them. The truth though was that the introduced practices conflicted with the way these members traditionally carried on their daily activities. They felt pressured to conform to the new practices.

In this study, change in the structure of the organization was highly recommended prior to introducing the agile practices for the first time. With the current waterfall like operational structure, it was clear that the agile practices will not necessarily fit in every development activity that is being performed. The company should have started by assessing their existing processes and identifying areas where potential conflicts could arise.

Also consulting with the different project members is a key factor to the success of the adoption process. The members should have been exposed to the concepts of the agile methodologies prior to applying them and asked whether or not these practices make sense to them and fit their daily activities. Their input would have been very valuable to tailor the introduced practices to fit their expectation. Also the sense of being included in the decision making process could have positively affected the members' willingness to accept the new agile practices. Redefining the members' roles and restructuring the teams based on the process assessment and the members' feedback would have greatly altered the way the adoption process was perceived.

This study concludes that in any organization, prior assessment of the organizational structure as well as the practices that will be introduced is essential. The assessment of the new practices will lead the organization to understanding their underlying requirements that are needed to implement them. At this point, it can be determined whether these practices fit the existing structure of the organization or not. The popularity of certain practices or methodologies should not be the only factor that organizations use to make decisions about their adoption. The fit of the introduced practices is a major factor that could make or break the adoption process. A given set of practices could work perfectly in one environment, yet fail in another.

When a misfit between the structure of an organization and the practices it adopts occurs, many symptoms of this misfit can be observed. It is very likely to observe difficulties in communication between teams, lots of conflicts and misunderstandings, collaboration difficulties, and more. It becomes a struggle to adhere to these practices when they conflict in various ways with the existing organizational structure. It becomes even harder when these practices conflict with the day-to-day work that the members of the organization have to perform. It is recommended to spend some time upfront to understand the requirements needed to implement the new practices prior to adopting them, rather than spending more time after the adoption process to fix the problems these practices may create.

8. Future Research

The limitations of this study along with the concluded results uncovered potential opportunities for various future research initiatives. A possible research area could be to examine an organization and the interaction between its members before and after introducing agile methodologies. The researchers can then easily compare the before and after snapshots of the organization and the relationships between its members. This may provide them with more precise conclusions and may allow them to identify and attribute any observed behavioral patterns to the appropriate cause.

Another potential research area could be to examine organizations where their structure was altered to accommodate the changes offered by the agile methodologies. The organizational structure alterations could be done by changing the existing processes. It can also be introduced through redefining the roles of the members involved in agile projects. It will be interesting to observe the perception of the projects members in this scenario, and see if they will be able to easily adapt to the agile practices or not.

Finally, a potential research area could be to examine how organizations deal with the limitations they have around their highly specialized resources,. An example of these resources is individuals with specific technical abilities that must be shared across different development projects. The researchers can examine how these resources are managed and assigned to different projects. They can also assess the impact of the high demand of these resources on the other projects members as well as on the resources themselves. The findings of this research may possibly be able to provide guidelines for organizations to facilitate managing and utilizing their highly specialized resources.

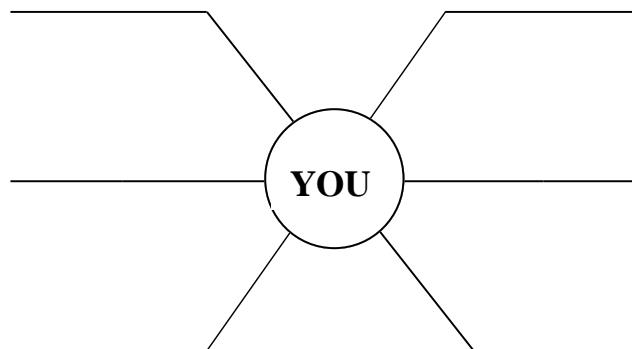
Appendix A - Interview Questions

Background/ Overview of job:

- 1) What is your official position or job title?
- 2) How long have you been in this position? How long have you been with the company?
- 3) Could you briefly describe the main types of activities you do in your job?

Project Overview

- 4) Could you describe your role/interaction in the Route Service project?
 - At what point did you get involved?
 - How long were you involved for?
 - Work load (number of tasks/ number of sprints)?
- 5) Using the diagram, could you please identify the other people / teams you interacted with during this project
 - Ask the participants to define the key links
 - Start with the most important ones and work your way through the rest as time permits
 - Try to ask groups that are not involved in the main conflict/problem to provide their input on how they view the relationship between the two main teams (show the network map to groups and ask them to comment on other teams' relationships)



Network Interactions

- 6) Could you give some specific examples of the kinds of things the other teams do/did that were helpful to you in this project?
- 7) Could you give some specific examples of the kinds of things the other teams do/did that were not helpful to you in this project? How did this impact your job?

Ask the following questions explicitly if this issue doesn't come up in the conversation

Problem Overview

- 8) Are you aware of any current conflicts between teams involved in this project? (Description of the problem)
- 9) Why do you think this problem occurred?
- 10) What could have been done to avoid this problem?
- 11) How do you see this problem getting resolved?
- 12) Lessons learned from this project/problem?

Follow up with the following questions if they did not get answered during the discussion of questions 5, 6, and 7

- 13) How much time was spent on this project (daily/weekly)? Is it the same as the allocation time specified in the sprints? If different, why?
- 14) How much time was spent on attending meetings (stand ups, retrospectives, demos, document reviews)?
- 15) What percentage of this time do you personally consider beneficial? What percentage is considered overhead?
- 16) Do you think you have a good understanding of the project scope? Was this the case from the first time you joined the project or was it developed over time?
- 17) Do you think the project scope is too big? Was it supposed to be broken down into smaller "sub-projects"?
- 18) Why do you think that the project was extended beyond the initial delivery time? What could have been done to avoid this? (behavior / methodology problems)

Appendix B- Favorable and Unfavorable Aspects of an Organizational Culture

Dimension	Favorable aspects in relation to XP	Unfavorable aspects in relation to XP
Innovation and Risk	<ul style="list-style-type: none"> • Innovative and creative developers • Managers open to suggestions • Rewarding system which stimulates creativity and performance • Errors dealt with as part of the learning process 	<ul style="list-style-type: none"> • Developers with a conservative profile • Managers closed to suggestions • Rewarding system stimulating only following orders • Error punishment system
Detail Orientation	<ul style="list-style-type: none"> • Analytical developers • Philosophy of constant improvement • Application of tests and reviews is important 	<ul style="list-style-type: none"> • Attention to the whole • Resistance to testing and reviewing intensification • Lack of quality control
Outcomes Orientation	<ul style="list-style-type: none"> • Valuation of employees • Concern with the customer's satisfaction • Availability of material and intellectual resources to projects 	<ul style="list-style-type: none"> • Strong orientation to profitability and cost reduction in detriment to employees and customers' valuation • Demand developers' performance in terms of extra hours
People Orientation	<ul style="list-style-type: none"> • Motivation • Benefits • Satisfactory salary • Career plan • Good relation between the development team and the management 	<ul style="list-style-type: none"> • Discouraged developers • Absence of human capital valuing policies • Poor relation between the development team and the management
Team Orientation	<ul style="list-style-type: none"> • Team-wise tasks • Team-oriented rewarding system • Team-oriented evaluative system • Team spirit and cohesion • Balanced skills, experiences and personalities among developers 	<ul style="list-style-type: none"> • Individual-oriented tasks • Individual-oriented rewarding system • Individual-oriented evaluative system • Lack of cohesion in the development team • Unbalanced skills, experiences and personalities among developers
Aggressiveness	<ul style="list-style-type: none"> • Collaboration • Knowledge and information 	<ul style="list-style-type: none"> • Competition • Formality

	sharing <ul style="list-style-type: none"> • Trust and informality • Solving conflicts democratically 	<ul style="list-style-type: none"> • Personal conflicts • Solving conflicts through imposition
Stability	<ul style="list-style-type: none"> • Open to new technologies • Open to new ways to work 	<ul style="list-style-type: none"> • It is limited to known technologies even without satisfactory results • It seeks for security in routine techniques even without satisfactory results

References

- [1] K.Schwaber and M. Beedle, “Agile Software Development with SCRUM”, Prentice Hall, Upper Saddle River, 2001.
- [2] J. Argenti, “Corporate collapse: the causes and symptoms”, 1976.
- [3] T. Dyba, T. Dingsoyr,” Empirical studies of agile software development: A systematic review”, 2008
.
- [4] A. Cockburn, “Crystal Clear: A Human-Powered Methodology for Small Teams”, Addison-Wesley, ISBN 0-201-69947-8, 2004.
- [5] J. Stapleton, “DSDM: Business Focused Development”, second ed., Pearson Education, ISBN 978-0321112248, 2003.
- [6] S.R. Palmer, J.M. Felsing, “A Practical Guide to Feature-driven Development”, Prentice Hall, Upper Saddle River, NJ, ISBN 0-13-067615-2, 2002.
- [7] M. Poppendieck, T. Poppendieck, “Lean Software Development – An Agile Toolkit for Software Development Managers”, Addison-Wesley, Boston, ISBN 0-321-15078-3, 2003.
- [8] K. Beck, “Extreme Programming Explained: Embrace Change”, Addison-Wesley, ISBN 0-201-61641-6, 2000.
- [9] K. Beck, “Extreme Programming Explained: Embrace Change”, second ed., Addison-Wesley, ISBN 978-0321278654, 2004.
- [10] B. Bahli, E.S.A. Zeid, “The role of knowledge creation in adopting extreme programming model: an empirical study”, ITI 3rd International Conference on Information and Communications Technology: Enabling Technologies for the New Knowledge Society, 2005.

- [11] R. Baskerville, B. Ramesh, L. Levine, J. Pries-Heje, S. Slaughter, "Is internet-speed software development different?", *IEEE Software* 20(6), 70–77, 2003.
- [12] M. Ceschi, A. Sillitti, G. Succi, S. De Panfilis, "Project management in plan-based and agile companies", *IEEE Software* 22(3), 21–27, 2005.
- [13] J. Chong, Social behaviors on XP and non-XP teams: a comparative study, in: *Proceedings of the Agile Development Conference (ADC'05)*, 2005.
- [14] A. Dagnino, K. Smiley, H. Srikanth, A.I. Anton, L. Williams, "Experiences in applying agile software development practices in new product development", *Proceedings of the 8th IASTED International Conference on Software Engineering and Applications*, November 9–11, Cambridge, MA, United States, 2004.
- [15] D. Dalcher, O. Benediktsson, H. Thorbergsson, "Development life cycle management: a multi-project experiment", *Proceedings of the 12th International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05)*, 2005.
- [16] M.-R. Hilkkä, T. Tuure, R. Matti, "Is extreme programming just old wine in new bottles: a comparison of two cases", *Journal of Database Management* 16(4), 41–61, 2005.
- [17] S. Ilieva, P. Ivanov, E. Stefanova, "Analyses of an agile methodology implementation", *Proceedings 30th Euromicro Conference*, IEEE Computer Society Press, pp. 326–333, 2004.
- [18] D. Karlström, P. Runeson, "Combining agile methods with stage-gate project management", *IEEE Software* 22(3), 43–49, 2005.
- [19] J. Koskela, P. Abrahamsson, "On-site customer in an XP project: empirical results from a case study", T. Dingsøyr (Ed.), *Software Process Improvement, Proceedings, Lecture Notes in Computer Science*, vol. 3281, Springer-Verlag, Berlin, pp. 1–11, 2004.

- [20] L. Layman, L. Williams, L. Cunningham, “Exploring extreme programming in context: an industrial case study”, Agile Development Conference, 2004.
- [21] F. Macias, M. Holcombe, M. Gheorghe, “A formal experiment comparing extreme programming with traditional software construction”, Proceedings of the Fourth Mexican International Conference on Computer Science (ENC 2003), 2003.
- [22] A. Mackenzie, S. Monk, “From Cards to Code: How Extreme Programming Re-Embodies Programming as a Collective Practice”, Computer Supported Cooperative Work, vol. 13, pp. 91–117, 2004.
- [23] C. Mann, F. Maurer, “A case study on the impact of scrum on overtime and customer satisfaction”, Agile Development Conference, 2005.
- [24] K. Mannaro, M. Melis, and M. Marchesi, “Empirical analysis on the satisfaction of IT employees comparing XP practices with other software development methodologies”, Extreme Programming and Agile Processes in Software Engineering, Proceedings, Lecture Notes in Computer Science, vol. 3092, Springer Verlag, pp. 166–174, 2004.
- [25] A. Martin, R. Biddle, J. Noble, “The XP customer role in practice: three studies”, Agile Development Conference, 2004.
- [26] A. Martin, R. Biddle, J. Noble, “When XP met outsourcing, in Extreme Programming and Agile Processes in Software Engineering”, Proceedings, Lecture Notes in Computer Science, vol. 3092, Springer Verlag, Berlin, pp. 51–59, 2004.
- [27] G. Melnik, F. Maurer, “Perceptions of agile practices: a student survey”, Proceedings, eXtreme Programming/ Agile Universe 2002, Lecture Notes in Computer Science, vol. 2418, Springer Verlag, pp.241–250, 2002.
- [28] G. Melnik, F. Maurer, “A cross-program investigation of student’s perceptions of agile methods”, International Conference on Software Engineering (ICSE), St. Louis, MI, USA, 2005.

- [29] P. Middleton, Lean software development: two case studies, *Software Quality Journal* 9(4),241–252, 2001.
- [30] H. Robinson, H. Sharp, “The characteristics of XP teams, *Extreme Programming and Agile Processes in Software Engineering*”, *Lecture Notes in Computer Science*, vol. 3092, Springer Verlag, Berlin, pp. 139–147, 2004.
- [31] H. Robinson, H. Sharp, “The social side of technical practices, *Extreme Programming and Agile Processes in Software Engineering*”, *Lecture Notes in Computer Science*, vol. 3556, Springer Verlag, Berlin, pp. 100–108, 2005.
- [32] H.S. Robinson, “Organizational culture and XP: three case studies”, *Proceedings of the Agile Conference (ADC’05)*, 2005.
- [33] H. Sharp H. Robinson, “An ethnographic study of XP practice”, *Empirical Software Engineering*, 9(4), 353–375, 2004.
- [34] A. Sillitti, M. Ceschi, B. Russo, G. Succi, “Managing uncertainty in requirements: a survey in documentation- driven and agile companies”, *Proceedings of the 11th International Software Metrics Symposium (METRICS)*, 2005.
- [35] H. Svensson, M. Höst, “Views from an organization on how agile development affects its collaboration with a software development team”, *Lecture Notes in Computer Science*, vol. 3547, Springer Verlag, Berlin, pp. 487–501, 2005.
- [36] H. Svensson,M. Höst, “Introducing agile process in a software maintenance and evolution organization”, *Ninth European Conference on Software Maintenance and Reengineering (CSMR’05)*, 2005.
- [37] Bjornar Tessem,, “Experiences in learning xp practices: a qualitative study”, *XP 2003*, vol. 2675, Springer Verlag, Berlin, pp. 131–137, 2003.

- [38] C.A. Wellington, T. Briggs, C.D. Girard, “Comparison of student experiences with plan-driven and agile methodologies”, Proceedings of the 35th ASEE/ IEEE Frontiers in Education Conference, 2005.
- [39] S.M. Young, H.M. Edwards, S. McDonald, J.B. Thompson, “Personality characteristics in an XP team: A repertory grid study”, Proceedings of Human and Social Factors of Software Engineering (HSSE), St. Louis, MI, USA, 2005.
- [40] Noura Abbas, A. M. Gravell, and Gary B. Wills, “Historical Roots of Agile Methods: Where Did “Agile Thinking” Come From?”, Agile Processes in Software Engineering and Extreme Programming Lecture Notes in Business Information Processing, Volume 9, Part 4, 94-103, 2008.
- [41] Larman, C., Basili, V.R.: Iterative and Incremental Development: A Brief History. IEEE Computer Society 36(6), 47–56, 2003.
- [42] Beck, Kent, et al. "Manifesto for Agile Software Development", can be found online at <http://agilemanifesto.org>, 2001.
- [43] K. Petersen, Blekinge Institute of Technology Doctoral Dissertation Series, No 2010:04, ISBN 978-91-7295-180-8, 2010.
- [44] K. Petersen and C. Wohlin, “A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case”, Journal of Systems and Software, 82(9):1479–1490, 2009.
- [45] J. B. Barlow, J. S. Giboney, M. J. Keith, D. W. Wilson, R.M. Schuetzler, P. B. Lowry, and A. Vance, CAIS Journal, Vol.29, 2011.
- [46] Gilb T, "Evolutionary Delivery versus the "waterfall model" " ACM SIGSOFT Software Engineering Notes 10(3): 49-61, 1985.
- [47] Martin J., “Rapid Application Development”, Macmillan Publishing Co., Inc., 1991.

- [48] Raymond, E.S., “The Cathedral and the Bazaar”, 1999.
- [49] Saint-Exupery, A.d., “Wind, Sand and Stars”, Harcourt, 1992.
- [50] Turner, R. and A. Jain, “Agile Meets CMMI: Culture Clash or Common Cause?”, XP/Agile Universe LNCS 2418 p. 153-165, 2002.
- [51] Weinberg, G.M., “The Psychology of Computer Programming”, John Wiley & Sons, Inc. 304, 1985.
- [52] Yourdon, E., “Death March: The Complete Software Developer's Guide to Surviving Mission Impossible Projects”, ed. D.B. Paul, Prentice Hall PTR. 227, 1997.
- [53] Dijkstra, E.W., “The humble programmer”, Communication of the ACM, 15(10, p. 859-866, 1972.
- [54] C. Tolfo, R. S. Wazlawick, “The influence of organizational culture on the adoption of extreme programming”, Journal of Systems and Software, Volume 81, Issue 11, Pages 1955–1967, November 2008.
- [55] J.Ivaria, N. Ivaria,, “The relationship between organizational culture and the deployment of agile methods”, MIS quarterly journal, Volume 31, Issue 1, Page 35-58, March 2007.
- [56]A. Cockburn, “Agile Software Development”, Addison-Wesley, ISBN 0-201-69969-9, 2001.

Glossary

Burn-down Chart: A visible chart, which indicates on a daily basis the amount of work remaining in the Sprint. It could be used to track the remaining work for the entire product as well

Impediments: Anything that prevents the team from meeting their potential. It is the scrum master's responsibility to eliminate it unless it is internal to the team. It becomes then the team's responsibility to address them.

Product Backlog: A prioritized list of stories that are waiting to be worked on.

Product Backlog Item: Any item that is on the backlog list, which will include user stories and possibly technical stories.

Product Owner: A person who holds the vision for the product and is responsible for maintain, prioritizing, and updating the product backlog.

Project Team: The team responsible for committing to work, delivering and driving the product forward from tactical perspective

Retrospective: A session where the team and the scrum master reflect on the process and make commitments to improve

Scrum: A simple low ceremony planning approach.

Scrum Master: a servant leader to the team, responsible for removing impediments and making sure the process runs smoothly so the team can be as productive as possible

Sprint: A short (2-4 week) development cycle focused on delivering an increment of useful business functionality.

Sprint Planning: A meeting between the team and the product owner to plan the sprint and arrive at an agreement on the commitment.

Sprint Task: A single small item of work that helps one particular story reach completion

Stakeholder: Anyone external to the team with an interest in the product being developed.

Stand-up: A fifteen minute daily meeting to share progress, report impediments, and make commitments.

Task List: The tasks need to complete the set of stories committed to a sprint.